UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Solução Integrada de Video-Based Vehicle Identification

Eduardo Filipe de Lima Borrego

Mestrado em Engenharia Informática
Especialização em Sistemas de Informação

Trabalho de Projeto orientado por:
Prof. Doutor Luís Manuel Ferreira Fernandes Moniz
e Eng. António Ricardo Ruano Pinto

2018

Agradecimentos

Antes de mais quero agradecer à minha família. Se não fosse o apoio constante ao longos destes 5 anos, seja ao nível emocional como monetário, nunca poderia ter chegado a este ponto. Ter de ouvir inúmeras vezes conceitos relacionados de informática e programação quando não são da mesma área é preciso ter muita paciência.

Tenho de agradecer aos meus amigos e colegas (e eles sabem muito bem quem são) que sempre estiveram disponíveis para me ajudar. Sem eles muito provavelmente não teria chegado aqui. Nunca lhes poderei retribuir todo o tempo que despenderam para me ajudar.

Também tenho de agradecer à Accenture por me ter dado a possibilidade de realizar o projeto de mestrado num ambiente empresarial. Concederam-me todas as condições para poder realizar o projeto.

A equipa de *Tolling*, onde fui inserido, esteve sempre disponível para me ajudar. Fui muito bem-recebido por toda a equipa, e a convivência e proximidade dos vários elementos da equipa contribuiriam para um bom ambiente.

E claro, tenho de agradecer aos meus orientadores Luís Moniz, António Pinto e Francisco Antunes. O Francisco foi o que precisou de mais paciência para me aturar, visto que o chateava quase todos os dias. O António deu me total liberdade para gerir o tempo do horário de trabalho, agradeço imenso por isso! E por fim, agradeço imenso ao Luís Moniz por me ter ajudado em alturas fulcrais da tese e que contribuíram muito para uma melhor compreensão do problema em mãos.

Aos meus pais, Benedita e Filipe,

À minha irmã, Raquel.

Resumo

O congestionamento de tráfego é hoje em dia um dos problemas que mais aflige as grandes cidades. Ao longo dos anos verificou-se um crescimento do volume de tráfego que excede claramente as capacidades físicas das infraestruturas rodoviárias, resultando num crescimento exponencial do congestionamento de tráfego. Existem inúmeras técnicas para gerir o congestionamento de tráfego, mas nenhuma é capaz de prever condições de tráfego no futuro.

É neste sentido que surge o projeto de tese em que aborda a temática de *Dynamic Pricing* no setor das portagens. Como tal, o meu projeto consistiu em desenvolver um serviço de *dynamic pricing* baseado num modelo de aprendizagem para a atribuição de preços nas portagens, com base em métricas que são recolhidas em tempo real. Tendo em conta que o projeto se realizou no contexto de um projeto da Accenture dedicado ao sector das portagens, permitiu uma melhor compreensão do problema em questão. A informação fundamental para o desenvolvimento do serviço são as transações geradas nos pórticos das portagens. Adicionalmente, consideraram-se eventos meteorológicos, desportivos e feriados.

Com este projeto foi elaborado um mecanismo que maximize o lucro obtido na Illinois Tollway, incentive os condutores a optarem por outras rotas, não só para descongestionar alguns troços de estrada, como também diminuir o tempo perdido em filas de trânsito e, por sua vez, melhorar a qualidade de vida dos condutores.

Deste modo, o objetivo deste projeto pode-se subdividir em 3 partes: Recolha e tratamento de grandes quantidades de dados; Criação e treino de um modelo de aprendizagem; e Aplicação de um algoritmo para a atribuição de preços dinamicamente. O tratamento de dados e a criação e treino do modelo de aprendizagem foram realizados a partir de uma *framework* dedicada a *big data* e *machine learning*, de modo a otimizar o processo. A análise do congestionamento de tráfego ocorreu a partir da atribuição de um nível de serviço.

Palavras-chave: *Congestion Pricing*, *Dynamic Pricing*, *Machine Learning*, *Big Data* e Otimização do problema.

Abstract

Nowadays, the traffic congestion is one of the problems that disturbs major cities. Over the years there has been an increase in traffic volume which clearly exceeds the physical capacity of road infrastructures, resulting in an exponential increase in traffic congestion. Given the lack of effective solutions the need arose to create a new one to combat this problem.

That is how my project came up, and aproaches the subject of Dynamic Pricing in the toll industry. So, the project project consisted in developing a service for the dynamic atribution of prices using a machine learning technique to create a knowledge model to predict them, based on metrics that are collected in real time. Taking into account that the project was carried out in the context of an Accenture project on the tolls sector, it allowed a better understanding of the problem in question. The key information for the development of the service is the transactions generated on plazas.

This project intends to develop a mechanism to maximize the profit obtained in Illinois Tolway, encourage drivers to choose other routes, not only to minimize the traffic congestion of some sections of road, but also to reduce the time lost in traffic queues and improve the quality of life of drivers.

Therefore, the purpose of my project can be subdivided into 3 distinct parts: Collection and Processiong of large amounts of data; Creation and Training of a Knowledge Model; and Application of an algorithm for the assignment of prices dynamically. The data processing as the creation and training of the learning model will be carried out from a framework dedicated to Big Data and Machine Learning, in order to optimize the process. The analysis of traffic congestion will occur from the assignment of a service level.

Keywords: Dynamic Pricing, Traffic Congestion, Traffic Optimization, Machine Learning and Big Data

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste capítulo são abordados os temas de congestionamento de tráfego automóvel com ênfase nas técnicas de *dynamic pricing* nas portagens, uma visão geral de *Big Data* e dos algoritmos de *machine learning* mais utilizados hoje em dia. Estes temas são a base do Projeto de Engenharia Informática (PEI) que consiste em desenvolver um sistema de *Dynamic Pricing* através da aplicação de um algoritmo de *machine learning*. Este capítulo contém as seguintes componentes: Motivação, Objetivos, Instituição de Acolhimento, Planeamento e Estrutura do documento.

1.1 Motivação

Ao longo dos anos tem-se verificado um crescimento do volume de tráfego que não é acompanhado pelo investimento nas infraestruturas rodoviárias, resultando num crescimento exponencial da congestão de tráfego. A congestão interrompe o fluxo de tráfego e atrasa o movimento dos cidadãos. Por sua vez, implica mais tempo despendido no trânsito, mais custos em combustíveis e mais emissões nocivas para o ambiente. Além disso, o veículo tem tendência a sofrer maior desgaste. Face a estes problemas é necessário criar mecanismos que visem melhorar o fluxo de tráfego.

A resposta tradicional a este problema é a expansão das capacidades rodoviárias, mas que tem custos associados muito elevados e nem sempre é possível pela própria topologia do terreno ou por questões ambientais [1].

Outra medida que visa aliviar a congestão de tráfego é o *Congestion Pricing*, ou seja, é a cobrança pela circulação de determinadas vias, na tentativa de alterar rotas mais críticas para outros meios de transporte ou percursos alternativos, durante as horas de maior tráfego.

Nos últimos anos, com os desenvolvimentos tecnológicos nos sistemas eletrónicos de cobrança de portagens, a atribuição de preços pode ocorrer de forma dinâmica, ou seja, as portagens podem ser configuradas em tempo real de acordo com um conjunto de condições de tráfego. É neste sentido que surge o conceito de *Dynamic Pricing*, ou

seja, os preços alteram-se dinamicamente ao longo do tempo com base em fatores como a densidade de tráfego, condições meteorológicas e até situações de acidentes.

Embora já existam diversos sistemas de portagens que aplicam *dynamic pricing*, não existem sistemas que aprendam a tomar novas decisões com base no seu histórico de decisões. Deste modo, o objetivo deste PEI consiste em desenvolver um sistema *Dynamic Princing* baseado num modelo de aprendizagem para a atribuição de preços nas portagens. O desempenho do sistema irá depender da framework de *Big Data* escolhida para o efeito, e da estratégia holística de gestão de informação, de modo a permitir a análise de grandes quantidades de informação [**?**]. Outro ponto a considerar é o facto do sector das portagens envolver milhões de transações por dia, tornando-se imprescindível o uso de um sistema de *Big Data*. Por sua vez, será necessário descobrir padrões e correlações de dados para escolher o algoritmo de *machine learning* ideal. Em caso de sucesso do desenvolvimento deste sistema será possível implementá-lo numa infraestrutura rodoviária constituída por diversas auto-estradas.

Algo muito importante a ter em consideração é que, a atribuição dinâmica de preços não se resume a obter mais lucro. O grande objetivo é alterar os hábitos dos condutores a longo prazo. Como tal, quando se pretende incentivar um condutor a mudar a sua rota, é imprescindível que traga vantagens para o mesmo, tais como: a redução de custos, viagens mais rápidas, previsíveis e de fácil acesso.

Os pontos fulcrais para aceitação do sistema são

1. A prática de preços justos e realistas para a situação em questão;
2. Os preços não devem ser atualizados em tempos inferiores a 15 minutos de modo a que os condutores consigam preparar as sua viagens com antecedência;
3. Garantir a transparência dos preços aplicados;

1.2 Objetivos

Tendo em conta o que foi exposto na secção anterior, pretende-se alcançar os seguintes objetivos:

- **Desenvolvimento de um serviço de *dynamic pricing* baseado num modelo de aprendizagem para a atribuição de preços nas portagens**
  Consiste em desenvolver um sistema capaz de prever métricas que permitam a atribuição de novos preços para cada classe de veículo num momento futuro, com base num modelo de aprendizagem. Para tal, é necessário ter em conta os seguintes pontos:

  1. **Recolha e tratamento de grandes quantidades de dados utilizando Big Data**
     Tendo em conta ao elevado número de veículos que circulam nas auto-estradas,

são geradas milhões de transações por dia. É neste sentido que surge a necessidade da aplicação de uma estratégia holística de gestão de informação, de modo a permitir a análise de grandes quantidades de informação, e, por sua vez, descobrir padrões e correlações de dados. É necessário assim determinar qual é o método de processamento de dados ótimo para o problema em questão.

2. **Criação de um modelo de aprendizagem**
   Como se pretende que o sistema seja capaz de aprender a tomar decisões, é estritamente necessário desenvolver um modelo que se baseie em dados de treino com informações do estado de congestionamento de tráfego numa determinada auto-estrada do último ano. Como já existem implementações de algoritmos de ML, o algoritmo ideal será o que resolva o problema de forma mais eficiente.
3. **Aplicação de um algoritmo para a atribuição de preços dinamicamente**
   Tendo por base o modelo de aprendizagem criado anteriormente, é necessário criar uma métrica que seja capaz de devolver uma informação essencial para analisar e atribuir uma representação abstrata do estado de tráfego e, por sua vez, atualizar os preços das tarifas de cada classe de veículo de um determinado momento no futuro.

1.3 Instituição de acolhimento

Este PEI foi realizado na Accenture, uma consultora multinacional "líder em serviços profissionais que oferece uma ampla gama de serviços e soluções em estratégia, consultoria, digital, tecnologia e operações". Hoje em dia tem clientes em mais de 120 países, em mais de 40 indústrias e conta com mais de 400 mil funcionários. A empresa está dividida em 5 partes: Accenture *Strategy*, Accenture *Consulting*, Accenture *Digital*, Accenture *Technology* e Accenture *Operations*.

O projeto onde fui integrado chama-se Centro de Excelência de Tolling da Accenture, o qual faz parte da Accenture Technology, e é constituído por uma equipa especializada na área de *Tolling*. Dedica-se ao desenvolvimento, implementação e manutenção de 4 componentes:

- **Componente pública** (*Corporate*)
- **SSW** - *Self-service Website*
- **OBO** - *Operational Back-Office*
- **CBO** - *Commercial Back-Office*

É suportada pelo *Lisbon Delivery Center* (LDC), com mais de 200 consultores, que fornece serviços a clientes de 14 países. Um dos seus clientes nacionais é a Ascendi. A

nível internacional tem como um dos seus clientes mais importantes, a *Illinois State Toll Highway Authority* (ISTHA), ou simplesmente *Illinois Tollway*, para a qual fornece o maior sistema de *Tolling Back Office* dos Estados Unidos da América (USA). Para o desenvolvimento do PEI serão disponibilizados dados reais gerados nos pórticos das portagens da *Illinois Tolway* e, fui orientado pelo Eng. António Pinto, *Deliver Lead* e pelo Eng. Francisco Antunes, *Batch Lead*.

1.4 Planeamento

	Atividades	Data Início	Data Fim	Observações
1	***Análise***	**30-09-2017**	**18-12-2017**	
1.1	Entendimento do problema de negócio	30-09-2017	30-11-2017	
1.1.1	Identificação dos requisitos de negócio e operacionais	08-10-2017	30-11-2017	
1.1.2	Documentação de requisitos e desenho de alto nível da solução	30-11-2017	08-12-2017	
1.1.3	Relatório Preliminar de Estágio	29-10-2017	30-11-2017	
1.2	***Desenho***	**04-12-2017**	**26-01-2018**	
1.2.1	Desenho dos processos de negócio e operacionais	04-12-2017	08-12-2017	
1.2.2	Desenho da arquitetura funcional	11-12-2017	22-12-2017	
1.2.3	Desenho da arquitetura técnica	08-01-2018	12-01-2018	
1.2.4	Definição da estratégia de testes	15-01-2018	17-01-2018	
1.2.5	Documentação de Desenho Funcional e Técnico	18-01-2018	25-01-2018	
1.2.6	Documentação de Estratégia de Testes	25-01-2018	26-01-2018	
1.3	***Desenvolvimento e Testes***	**29-01-2018**	**31-05-2018**	
1.3.1	Setup do ambiente de desenvolvimento	29-01-2018	31-01-2018	
1.3.2	Codificação dos componentes	01-02-2018	30-03-2018	
1.3.3	Testes de Volume e Bug Fixing	02-04-2018	06-04-2018	
1.3.4	Testes de Stress e Bug Fixing	09-04-2018	13-04-2018	
1.3.5	Testes de *Throughput* e Bug Fixing	16-04-2018	20-04-2018	
1.3.6	Testes de Integração e Bug Fixing	23-04-2018	27-04-2018	
1.3.7	Testes de Sistema e Bug Fixing	30-05-2018	04-05-2018	
1.3.8	Documentação de Setup, Codificação e Testes	07-05-2018	11-05-2018	
1.4	**Preparação dos Resultados Estatísticos e Produção de Relatório de Estágio**	**14-05-2018**	**17-06-2018**	

Tabela 1.1: Planeamento do Projeto

Nota: embora que a produção do relatório tenha sido realizada no período estipulado, apenas foi entregue no fim do mês de Setembro. A revisão do mesmo acarretou algum tempo justificando o atraso da entrega final. Além disso, em meados de Abril tive de proceder a alterações tanto na implementação como na documentação. Como tal, o planeamento de projeto não reflete o trabalho desenvolvido.

1.5 Estrutura do documento

O relatório está dividido em 7 capítulos:

- **Capítulo 1 - Introdução**
  Neste capítulo evidenciam-se as motivações que levaram à realização do PEI, os objetivos a concretizar, uma breve exposição da instituição de acolhimento e o mapa da calendarização das tarefas a realizar.
- **Capítulo 2 - Trabalho relacionado**
  Neste capítulo realiza-se um levantamento das áreas de foco do PEI, evidenciando os sistemas e técnicas existentes, e a comparação respetiva para servir de base para uma decisão fundamentada das técnicas a aplicar no projeto.
- **Capítulo 3 - Análise**
  Neste capítulo evidenciam-se as decisões de implementação tais como: a estrutura do aplicação a desenvolver, a exposição do algoritmo de análise do estado de congestionamento de tráfego a ser implementado, e as ferramentas e metodologias a utilizar.
- **Capítulo 4 - Desenho**
  Neste capítulo evidenciam-se os requisitos de negócio e operacionais, o desenho de alto nível da solução, modelo de domínio, os *Key Performance Indicators* (KPIs) e os Casos de Uso.
- **Capítulo 5 - Implementação**
  Neste capítulo evidencia-se todo o processo de desenvolvimento do sistema detalhando em várias secções todos os passos do algoritmo.
- **Capítulo 6 - Testes e Análise ao Sistema**
  Neste capítulo evidenciam-se os testes realizados ao algoritmo, incluindo a sua identificação, descrição, resultados esperados e obtidos.
- **Capítulo 7 - Conclusões**
  Neste capítulo realiza-se um levantamento do trabalho realizado, incluindo as minhas contribuições e competências adquiridas, evidenciam-se as conclusões finais, as dificuldades que existiram ao longo do projeto, e possível trabalho futuro.

Capítulo 2

Trabalho relacionado

Neste capítulo são abordados os três temas principais deste PEI (*congestion pricing*, *big data* e *machine learning*), de modo a salientar os mecanismos e técnicas existentes.

2.1 *Congestion Pricing*

O aumento do volume de tráfego é algo que afeta severamente a qualidade de vida dos cidadãos. O pára-arranca constante, além de criar problemas mecânicos nos veículos, contribui para um consumo excessivo de combustível que acarretam não só custos monetários, como também promove o aumento de emissões nocivas para o ambiente. Face a estes problemas é necessário criar mecanismos que visem melhorar o fluxo de tráfego.

A resposta tradicional a este problema é a expansão das capacidades rodoviárias, que têm custos muito elevados e nem sempre é possível pela própria topologia do terreno ou por questões ambientais [1].

Uma segunda abordagem passa por limitar o acesso a determinados locais durante certos períodos do dia, como por exemplo, taxar os parques de estacionamento. Por exemplo, todos os dias entram em Lisboa cerca de 360 mil carros quando a cidade apenas tem estacionamento para 200 mil. Sob a alçada da EMEL (Empresa Municipal de Mobilidade e Estacionamento de Lisboa) há atualmente cerca de 54 mil lugares de estacionamento taxados, os quais irão aumentar para 84 mil. Esta medida provoca um grande descontentamento da população dado que perdem capacidades de circulação dentro da cidade [2].

Uma terceira hipótese, na qual que se insere o tema do projeto, consiste em melhorar a eficiência do sistema rodoviário, e ao mesmo tempo reduzir os custos. O *re-timing* dos semáforos e limitar o acesso às vias de acesso das autoestradas são exemplos da aplicação desta hipótese [1].

Outra medida que visa aliviar a congestão de tráfego é o *Congestion Pricing* que tem como objetivo cobrar a circulação de determinadas vias, na tentativa de incentivar os condutores a optarem por outros meios de transporte ou percursos alternativos, durante

as horas de maior tráfego. Desde sempre que os sistemas de pagamento nas autoestradas dependem das taxas sobre os combustíveis, de taxas especiais de consumo e de taxas de portagens. Esta metodologia tem como objetivo pagar os custos de construção, de manutenção e de operações nas autoestradas.

De acordo com a teoria microeconómica, a procura de um bem está diretamente relacionada com o preço do mesmo. Se a oferta de um bem for fixa, os preços podem ser aumentados quando a procura excede a oferta. Exemplos desta prática são os sistemas baseados no tempo para bilhetes de avião em fim-de-semana de férias, e para o uso diurno de telefones. Neste caso, os utilizadores que preferirem usufruir destes serviços no período de picos de congestionamento deverão possuir um serviço *premium*. O resto dos utilizadores irão escolher outra altura mais favorável, acarretando assim menos custos. Além disso, se mesmo com a prática de preços elevado, os utilizadores continuarem a aceder aos serviços, é um indicador que o serviço é rentável.

Embora esteja comprovado que o *Congestion Pricing* possa produzir soluções economicamente eficientes, pode prejudicar os condutores se as receitas não forem usadas para melhorar a mobilidade. Considere-se os seguintes exemplos:

- Alguns condutores irão preferir pagar mais e continuar a usar as mesmas vias, dado que nem sempre as alternativas propostas compensam o tempo de viagem. Tratam-se de condutores em que o tempo de viagem é um fator preponderante de circulação.
- Alguns condutores que anteriormente faziam determinadas rotas, irão diminuir a sua qualidade de viagem pela circulação de novos condutores. Vias que anteriormente fluíam sem perturbações poderão sofrer problemas de congestionamento [3];

Deste modo, uma tentativa para melhorar o fluxo de veículos passa por definir preços de tarifas que sejam adequados à severidade do congestionamento. Isto implica que as portagens devam variar de acordo com a hora do dia, classe do veículo, número de ocupantes, a localização e as condições em tempo-real tais como: volume de tráfego, as condições atmosféricas, acidentes e outros tipos de eventos excecionais [4]. Existem duas classes gerais de *Congestion Princing*: uma baseada em análises estáticas e outra baseada em análises dinâmicas. A primeira classe atribui sempre os mesmos preços independentemente das condições em tempo-real de trânsito. Por outro lado, a segunda classe tem em consideração um conjunto de fatores que podem influenciar o fluxo normal de trânsito.

Antes de entrar em detalhes na caracterização das abordagens de *Congestion Pricing*, é necessário compreender as estruturas físicas que compõem as autoestradas.

2.1.1 Tipos de Cobrança

A cobrança nas portagens ocorre através de pórticos (plazas), os quais podem ser de dois tipos: tradicionais ou eletrónicos.

Pórticos Tradicionais

Os pórticos tradicionais podem-se subdividir em 2 categorias: vias manuais assistidas por um colaborador da empresa ou vias manuais automáticas. Em ambos os casos, o meio de pagamento pode ser em dinheiro ou cartão [5].

Pórticos Eletrónicos

Os pórticos eletrónicos ocorrem através de *Electronic Toll Collection* (ETC) e visam eliminar os atrasos gerados nos pórticos tradicionais. A ETC determina se a passagem de um determinado veiculo está registada num determinado programa, alerta e debita eletronicamente as contas dos proprietários, sem que os mesmos tenham de parar a sua viagem. Os veículos estão equipados com dispositivos denominados *transponders* ou *tags*, que são detetados por antenas nas infraestruturas presentes nas estradas ou em pórticos [6].

Os sistemas ETC baseiam-se em 4 componentes: Identificação automática de veículos (AVI – *Automated Vehicle Identification*); Classificação Automática de Veículos (AVC – *Automated Vehicle Classification*); Processamento de Transações e *Violation enforcement*.

Identificação Automática de Veículos

Processo que determina a identidade de um veículo numa determinada portagem. Hoje em dia, a identificação ocorre através de uma antena que comunica com um *transponder* no veículo via comunicações dedicadas de curto alcance (DSRC – *Dedicated Short Range Communications*). As *tags* ou *transponders* são do tipo RFID (*Radio Frequency Identification*) e provaram ter uma precisão excelente sem que o condutor necessite de baixar a velocidade. A principal desvantagem é custo de aquisição do dispositivo [6].

Para evitar o uso de *transponders*, a identificação do veículo pode ocorrer através da identificação da matricula (ALPR – *Automatic License Plate Recognition*), ou seja, o veículo é identificado através da extração da matrícula através de um conjunto de imagens. A qualidade das imagens é o fator preponderante para o sucesso da ALPR [7]. O processo de extração de imagens ocorre nos *Road Side Equipment* (RES) junto dos pórticos, que são câmaras digitais com sensores de infravermelhos que permitem gravar uma imagem do veículo. Utilizam algoritmos de reconhecimento ótico de carateres (OCR) para identificar com precisão a matrícula [8].

Classificação Automática de Veículos

Processo relacionado com o ponto anterior. A maioria das portagens cobra tarifas diferentes para tipos de veículos diferentes. A forma mais simples de identificar o tipo de veículo é a sua classe. Esta informação é registada durante a AVI. O número de classes de veículos difere consoante a empresa responsável pelas portagens.

Processamento de Transações

Lida com a gestão de contas dos utilizadores, guardando todas as transações de portagens e dos pagamentos do utilizador na conta respetiva.

Violation enforcement

O *Violation Enforcement System* (VES) é útil para reduzir os casos de não pagamento das transações. Existem vários métodos, os quais não são de teor relevante para o contexto do

projeto.

2.1.2 *Congestion Pricing* – Abordagem Estática

Existem várias abordagens de implementação estática do *Congestion Pricing*. Os métodos mais utilizados são baseados na altura do dia, baseados na distância percorrida ou baseados em *cordon pricing*.

- **método baseado na altura do dia** - atribui taxas de circulação mais baixas fora das horas de ponta, de modo a reduzir a congestão de tráfego nesses períodos. O objetivo é desencorajar os cidadãos a circularem durante os períodos de maior tráfego. As taxas são pré-determinadas.
- **método baseado da distância percorrida** - caso simples em que as tarifas são aplicadas de acordo com o número de km/milhas percorridas.
- **método de *cordon pricing*** - aplicado geralmente por motoristas dentro das cidades. O serviço mais comum são os táxis [9].

Em Portugal, as portagens tendem a manter um valor constante independentemente da altura do dia, variando apenas na classe de veículo [6].

O mais comum é atribuir preços diferentes consoante a circulação ocorra de dia ou de noite. A Illinois Tollway para gerir a circulação de veículos pesados, atribui um preço de tarifa se a circulação ocorrer de dia ou de noite. Como os veículos pesados tendem a perturbar o fluxo normal de trânsito, as tarifas diárias são mais altas do que as tarifas noturnas, de modo a incentivar a circulação dos mesmos durante a noite [10]. Além disso, atribui um preço diferente consoante o método de cobrança. Se efetuar via *transponder* a tarifa a pagar é mais baixa do que se pagar em dinheiro.

Outro exemplo é o sistema de congestão em Estocolmo em que se taxou a entrada de veículos no centro da cidade. A medida permitiu uma redução do congestionamento de tráfego e melhorou a situação ambiental no centro da cidade [11].

Singapura foi a primeira cidade do mundo a implementar um sistema eletrónico de cobrança de portagens em todas as estradas que ligam ao centro da cidade. A taxa cobrada depende da localização do pórtico, hora do dia, altura do ano e tipo de veículo, a qual é atualizada quadrimestralmente e durante as férias escolares, de modo a manter uma velocidade alvo. Nas horas de ponta e durante os picos de congestionamento, as taxas variam durante um período máximo de trinta minutos para manter um fluxo constante de veículos [11].

Em Londres é aplicada uma taxa de congestionamento numa zona do centro da cidade para a maioria dos veículos, com o intuito de reduzir o tráfego e angariar fundos de investimento para o sistema de transportes da cidade. A taxa é aplicada nos dias úteis das 7h da manhã até às 18h com uma taxa diária de 11.50 libras. Esta medida teve um impacto direto na redução de 30% no volume de tráfego na zona delimitada, segundo um estudo

realizado em 2007. Por outro lado, ocorreu um crescimento do número de táxis, autocarros, e especialmente de bicicletas. A qualidade do ar melhorou significativamente com uma redução dos níveis de óxidos de nitrogénio em 13.4% entre 2002 e 2003 [12, 13].

2.1.3 *Congestion Pricing* – Abordagem Dinâmica

Antigamente, as empresas tornavam os preços de produtos/serviços fixos durante longos períodos de tempo. Isto deve-se principalmente à ausência de informação em tempo-real, custos de transação associados elevados e grandes investimentos para a implementação de software ou hardware de uma estratégia com *dynamic pricing*.

Preços dinâmicos também conhecidos por *take-it-or-leave-it prices*, são preços que se alteram dinamicamente ao longo do tempo com base em fatores como o tempo, a procura e a disponibilidade [14]. Pode-se aplicar o mesmo conceito no contexto do problema do congestionamento de tráfego.

A Uber, uma empresa prestadora de serviços eletrónicos na área do transporte privado urbano, atribui preços de viagens tendo em conta fatores como: as condições meteorológicas adversas, eventos desportivos ou realização de festas. Tendo um preço base, as influências destes fatores aumentam o preço final através de um fator multiplicativo [15].

Nos últimos anos, com os desenvolvimentos tecnológicos em sistemas eletrónicos de cobrança em portagens, a atribuição de preços pode ocorrer de forma dinâmica, ou seja, as portagens podem ser configuradas em tempo real, de acordo com um conjunto de condições de tráfego. Hoje em dia, a aplicação de preços dinâmicos apenas tem sido utilizada em *High Occupancy Toll* (HOT) [9].

Existem dois problemas gerais que degradam o desempenho dos sistemas de portagens com HOT:

- aplicação de algoritmos estáticos incapazes de lidarem com as propriedades dos sistemas de tráfego, podendo causar grandes atrasos;
- variações de fluxo de tráfego não expectáveis devido à falta de sensibilidade dos algoritmos aplicados.

O primeiro sistema efetivamente com *dynamic pricing* foi implementado em San Diego I-15 *FastTrak HOT lanes* em 1998. Os veículos com um único ocupante pagam portagens quando utilizam vias HOV (*High Occupancy Vehicle*). A tabela de tarifas varia dinamicamente a cada 6 min, dependendo do nível de congestionamento nas *express lanes* que mantêm sempre um LOS (*Level Of Service*) "C" para instalações HOT [9, 16]. Estudos posteriores indicaram que a implementação de *dynamic pricing* oferece um uso personalizado, logo os motoristas usam as *express lanes* quando mais necessitam ou quando lhes é mais benéfico. Outro estudo revelou que o tempo médio de viagem variava pouco nas *I-15 Express Lanes* desde que as condições de fluxo livre foram atendidas por alterações das tarifas em tempo-real [14].

Outro exemplo da aplicação de sistemas dinâmicos são as *Minnesota's I-394 HOT lanes*. Este sistema é mais complexo comparado com as *San Diego I-15 HOT lanes*, dado que existem múltiplos pontos de entrada e saída, e as tarifas de portagem são atualizados a cada 3 minutos, ao longo de 11 milhas. As tarifas aplicadas são baseadas nos níveis de serviço das *express lanes*. Similarmente ao exemplo anterior, o nível de serviço mínimo que o sistema pretende manter é o "C", ou seja, é expectável que não passem mais de 29 veículos num determinado ponto a cada 30 segundos e que a velocidade média esteja entre 50-55 mph. São colocados sensores a cada 0.5 milhas para determinar o nível de serviço [9].

O tema *dynamic pricing* aplicado a portagens foi abordado também na tese de mestrado de João Gomes, a qual foi realizada no mesmo projeto onde fui inserido. A tese de João Gomes consistia em delinear uma arquitetura que permitisse colecionar e explorar grandes quantidades de dados das passagens dos veículos junto dos pórticos, e analisar a utilização da rede de transportes ao longo de um dia. Existiu a necessidade de estudar e implementar um algoritmo de *pricing* que fosse robusto e capaz de atribuir valores realistas para as portagens. Teria que possibilitar também a extensão de ideia de *dynamic pricing* a qualquer topologia de uma autoestrada, de forma a ajustar-se aos vários objetivos de negócio pretendidos pelos operadores rodoviários [4]. O algoritmo foi implementado com relativo sucesso, dado que apenas tinha em consideração um fator de congestionamento (Jam Factor), o qual era fornecido por uma API (Here API), e definiu um intervalo de Jam Factor para cada nível de serviço (Level Of Service, LOS). Embora não tenha sido utilizado para a solução final, o cálculo de densidade, velocidade média e taxa fluxo foram implementados no projeto de tese. Tendo em conta à semelhança do seu projeto e facto de ter acesso à implementação, irei utilizar o mesmo como ponto de partida. A formulação de todos os cálculos realizados têm como base os princípios descritos no livro "Traffic & Highway Engineering"de Nicholas J. Garber e Lester A. Hoel.

2.1.4 Comparação de Abordagens de *Congestion Pricing*

Tendo em conta o que foi exposto, a Tabela 2.1 evidencia os fatores que cada abordagem tem em consideração no que toca à atribuição de preços das tarifas aplicadas nas portagens.

	Abordagem Estática	Abordagem Dinâmica
Classe do Veículo	×	×
Altura do dia	×	×
Distância percorrida	×	×
Atualização automática dos preços	× *	×
Densidade de Tráfego		×
Velocidade média		×
Taxa de Fluxo de Tráfego		×
Meteorologia		×
Nº de Acidentes		×

* Na maioria dos casos, os preços apenas são atualizados manualmente em longos períodos de tempo.

Tabela 2.1: Comparação entre os tipos de abordagens: Estática e Dinâmica

2.1.5 Inovações envolvidas na gestão de tráfego

Ainda dentro do assunto do congestionamento de tráfego mas sendo o foco a gestão de tráfego, surge a Holanda, um dos países mais avançados em termos de redes de autoestradas do mundo inteiro. Através da emissão de mensagens várias e sinalização elec¡trónica ao longo de toda a rede permite uma gestão ótima do tráfego. A gestão de tráfego via sensores visa otimizar o fluxo de tráfego, para encontrar as necessidades adequadas de cada autoestrada através da análise dos dados recebidos e, a execução de medidas que reduzam o congestionamento e minimizar os tempos de resposta a incidentes. Esta recolha de dados é fundamental para realizar qualquer tipo de sistema de controlo de tráfego inteligente. Os sensores incluem ciclos indutivos, dispositivos de deteção de tráfego não intrusivo, câmaras de vídeo e processamento de imagens de vídeo.

Uma das tecnologias a consideradas para a gestão de tráfego, passa pelo uso de fibra ótica ao logo de toda a rede, com dois objetivos distintos: detectar a pressão e a vibração das passagens dos veículos [17].

2.2 *Machine Learning*

À medida que os dados crescem e tornam-se mais complexos, o problema de encontrar padrões é fundamental. O reconhecimento de padrões tem origem nas engenharias enquanto que a técnicas de *machine learning* (ML) evoluíram das ciências computacionais. Contudo, estas atividades podem ser vistas como duas faces do mesmo problema, dado que, em conjunto contribuem para um desenvolvimento substancial de *software*. Hoje em dia, os métodos de ML têm vindo a ganhar popularidade em setores como o reconhecimento de imagens, diagnósticos médicos, gestão de *emails* e música

ML pode ser definida como o processo de construir sistemas computacionais que automaticamente aprendem com a experiência [18].

De uma forma geral, ML pode ser classificada em *supervised* ML, *unsupervised* ML e *reinforcement learning*.

- ***supervised* ML** – técnicas de aprendizagem a partir de dados de treino. Os dados de treino consistem em pares de objetos de entrada (tipicamente vetores) e dos resultados esperados. Incluem problemas de regressão ou de classificação. Num problema de regressão, pretende-se prever resultados através da aplicação de uma função contínua. Num problema de classificação, por outro lado, pretende-se prever resultados através de uma função discreta. Neste caso sabemos à partida um conjunto de categorias discretas (o nome da classe do objeto de entrada) com as quais iremos prever a categoria de novos resultados. [18]
- ***unsupervised* ML**– técnicas onde os nomes das classes dos objetos de entrada não são usados. Neste caso não existe uma estrutura de dados. Como tal, não se sabe à partida qual será o aspeto dos resultados. Podem ser classificados como *clustering*, estimação de densidade ou visualização. No caso de *clustering* o objetivo é encontrar grupos de exemplos similares nos dados já recolhidos. No caso da estimação de densidade pretende-se determinar a distribuição de dados dentro do espaço de entrada. Por fim, no caso de visualização pretende-se projetar os dados de um espaço de alta dimensão para um espaço com duas ou três dimensões [18].
- ***reinforcement learning*** – técnica em que um agente deve aprender um determinado comportamento através de interações tentativa-erro num ambiente dinâmico, ou seja, com base em informações recolhidas do ambiente exterior, irá realizar tarefas ajustando o seu comportamento com base no *feedback* recolhido desse mesmo meio [19].

No contexto do tema principal, o *Congestion Pricing* pretende-se avaliar o estado da infraestrutura rodoviária e atribuir-lhe um nível de serviço, os quais são definidos de A a F. Isto significa que à partida existem um conjunto classes com as quais iremos organizar os dados. Portanto estamos perante um problema de classificação. Deste modo, apenas serão explorados em detalhe os métodos de classificação.

2.2.1 Algoritmos baseados em lógica

Árvores de Decisão

As árvores de decisão são estruturas para classificar dados com atributos comuns. Cada árvore de decisão representa a regra que categoriza os dados com base nesses atributos. Uma árvore de decisão é composta por nós, folhas e ramos. Um nó representa uma característica numa instância para ser classificada, cada ramo representa o valor que o nó pode assumir, e cada folha representa a decisão tomada. As instâncias são classificadas a partir do nó *root* e ordenadas baseado nos valores das características. Existem vários métodos para encontrar a característica que melhor divide os dados de treino tais como: o ganho de informação [20] e o índice gini [21, 22].

Técnicas baseadas em regras

As árvores de decisão podem ser traduzidas num conjunto de regras, criando uma regra por cada caminho da árvore desde a *root* até a uma folha da árvore [23]. Contudo, as regras podem ser induzidas diretamente dos dados de treino usando um conjunto de algoritmos baseados em regras. As regras de classificação podem ser representadas em classes na forma normal disjuntiva (FND). O objetivo é construir o conjunto mais pequeno de regras que seja consistente com os dados de treino [24].

2.2.2 Algoritmos baseados em Redes Neuronais

Redes Neuronais com uma camada única

Conjunto de valores de entrada de uma determinada característica e um peso associado são conectados num vector peso/valor previsto (tipicamente números reais entre -1 e 1. De seguida o perceptrão computa o *input* ponderado.

Redes Neuronais com multi-camadas

Os perceptrões multi-camada ou redes neuronais artificiais consistem num conjunto de processadores simples e muito interligados, denominados como neurónios, que são análogos aos neurónios biológicos do cérebro. Os neurónios são conectados por ligações fortes que passam o sinal de um neurónio para outro. Cada ligação tem um peso numérico associado. Cada neurónio recebe um conjunto de sinais de entrada através das suas conexões, mas apenas produz uma resposta. Os pesos são meios básicos em memórias de longo prazo nas redes neuronais. A rede neuronal aprende através de ajustamentos sucessivos desses mesmos pesos [25].

Rede de funções de base radial

Do Inglês, *Radial Basis Function* (RBF) *networks*, é uma rede de *feedback* de três camadas, em cada existe uma unidade oculta que implementa a função de base radial, e uma unidade que implementa o peso da soma das unidades ocultas de *output*. O procedimento de treino é geralmente dividido em duas etapas. Em primeiro lugar, a partir de algoritmos de *clustering* o centro e a largura das camadas ocultas são determinados. Em segundo

lugar, os pesos que conectam a camada oculta com a camada de *output* são determinados pelos algoritmos de decomposição de valores singulares (*Singular Value Decomposition*, SVD) ou da média mínima quadrada *(Least Mean Squared*, LMS) [21].

2.2.3 Algoritmos baseados em aprendizagem estatística

Ao contrário do caso das redes neuronais artificiais, as abordagens estatísticas são caraterizadas por terem um modelo probabilístico explícito, que fornece a probabilidade de uma instância pertencer a uma classe, em vez de atribuir uma classe simplesmente [21]. As redes Bayesian são os algoritmos probabilísticos mais representativos.

Classificadores Naive Bayes

As redes Naive Bayesian (NB) são os algoritmos mais simples em redes Bayesian, os quais são compostos por um grafo acíclico direto com apenas um nó pai (representa o nó não observado) e vários nós filhos (representam os nós observados), partindo do pressuposto de uma independência forte entre nós filhos, no contexto de um único nó pai. [10] Deste modo o modelo independente baseia-se na estimação de resultados. [26]

Redes Bayesian

Uma rede bayesian (BN) é um modelo gráfico para relações probabilísticas entre um conjunto de variáveis que representam características. A estrutura de uma BN é um grafo acíclico direto e os nós que o compõem têm uma correspondência de um para um com as características. Os arcos representam as influências entre características enquanto que a ausência dos mesmos codifica as independências. A tarefa de aprendizagem em BN passa por aprender a estrutura primeiro e determinar os seus parâmetros. Parâmetros probabilísticos são organizados num conjunto de tabelas, uma por cada característica, na forma de uma distribuição condicional local. Após a codificação de independências, é possível reconstruir a distribuição multiplicando as tabelas geradas. [21]

2.2.4 Aprendizagem baseada em instâncias

Algoritmos de *lazy-learning* são caraterizados pela introdução de atrasos que induzem ou generalizam o processo antes que a classificação seja executada. Requerem menos tempo computacional na fase de treino, ao contrário das árvores de decisão ou redes de Bayesian, mas requerem mais tempo computacional durante o processo de classificação. O algoritmo mais representativo é o *k-Nearest Neighbour* (kNN) [21].

k-Nearest Neighbour (kNN)

Baseia-se no principio que as instâncias dentro de um *data set* terão propriedades semelhantes com outras instâncias [27]. O algoritmo considera um número k de instâncias vizinhas com as quais queremos comparar para classificar a nova instância. Geralmente as instâncias podem ser representadas como pontos num dado espaço com n dimensões, em que cada dimensão corresponde a uma característica usada para classificar as mesmas. A

posição absoluta das instâncias é menos relevante que a distância relativa entre instâncias, a qual é determinada como uma métrica de distância. A métrica escolhida deve promover a minimização da distância entre instâncias similares da mesma classe. As métricas mais utilizadas são: Minkowsky, Manhattan, Chebychev, Euclidean, Camberra e Correlação de ranking de Kendall [21].

2.2.5 *Support Vector Machines*

As *Support Vector Machines* (SVM) resolvem o problema da separação de dados através da noção de "margem", uma por cada lado do plano que separa duas classes de dados. O objetivo é maximizar a distância entre duas classes de modo a diminuir a taxa de erro de classificação. Quando é possível separar linearmente duas classes, a divisão ótima do plano é relativamente simples através da minimização quadrática do separador do plano. Quando os dados são linearmente separáveis, os pontos que compõe a margem de cada classe correspondem aos pontos do vetor de suporte, e a solução é representada como a combinação linear desses pontos. Outros pontos são ignorados. Deste modo, a complexidade de um SVM não é afetada pelo número de características encontradas nos dados de treino (o número de vetores selecionados pelo algoritmo normalmente é reduzido). Assim, O SVM é adequado para tarefas de aprendizagem com um número elevado de características [21].

2.2.6 Comparação de algoritmos

Como foi possível verificar, as técnicas de *supervised* ML evidenciadas podem ser aplicadas em inúmeros domínios. A escolha do algoritmo ideal depende inteiramente das características que são relevantes para o projeto em questão. De seguida é evidenciada uma comparação entre os algoritmos apresentados tendo em conta um conjunto de características [21].

	Árvores de Decisão	Redes neuronais	*Naive Bayes*	kNN	SVM	Aprendizagem por regras
Precisão em geral	**	***	*	**	****	**
Velocidade de aprendizagem tendo em conta o número de atributos e o número de instancias	***	*	****	****	*	**
Velocidade de Classificação	****	****	****	*	****	****
Tolerância a valores em falta	***	*	****	*	**	**
Tolerância aa atributos irrelevantes	***	*	**	**	****	**
Tolerância a atributos redundantes	**	**	*	**	***	**
Tolerância a atributos com elevada interdependência	**	***	*	*	***	**
Lidar com atributos discretos/ binários/ contínuos	****	*** (não discreto)	*** (não contínuo)	*** (não discreto diretamente)	** (não discreto)	*** (não contínuo diretamente)
Tolerância a ruído	**	**	***	*	**	*
Lidar com *overfitting* perigoso	**	*	***	***	**	**
Tentativas para aprendizagem incremental	**	***	****	****	**	*
Explicação de transparência do conhecimento/ classificações	****	*	****	**	*	****
Manipulação de parâmetros do modelo	***	*	****	***	*	***

Tabela 2.2: Comparação de algoritmos de aprendizagem (**** representa o melhor e * o pior desempenho). Adaptado de (Kotsiantis, 2007) [21]

2.2.7 Bibliotecas de *machine learning*

Com o aumento da procura da aplicação de *machine learning* em sistemas surgiram inúmeras bibliotecas de *machine learning* e *data-mining* em Java, as quais oferecem um conjunto de algoritmos, entre as quais encontram-se: a Java-ML, a Weka e a Mulan.

A Java-ML (Java-Machine Learning) é orientada à aplicação de *machine learning* em programas. Como tal, as suas interfaces são restritas ao essencial e de fácil entendimento. Contém um extenso conjunto de técnicas baseado em similaridade e oferece técnicas de seleção. A biblioteca está construída em torno de duas interfaces principais: Dataset

e Instance. Em termos gerais os algoritmos oferecidos estão organizados nas seguintes categorias: *Clustering*, *Classification*, *Feature selection*, *Data filters*, *Distance measures*, *Utilities*. Incluem também várias medidas de distância, semelhança e correlação [28].

A Weka (*Waikato Environment for Knowledge Analysis*) é um conjunto de bibliotecas Java que implementa muitos algoritmos de ML e *data-mining* . Os métodos primários na Weka são os *classifiers* e que induzem um conjunto de regras ou árvores de decisão que modela os dados. Inclui algoritmos para aprender regras de associação e agrupamento de dados. Fornece ferramentas para pré-processamento de dados, ou filtros. O facto do software Weka ser escrito totalmente em Java facilita a disponibilidade de ferramentas de *data-mining*, independentemente da plataforma computacional utilizada [29].

A Mulan é uma biblioteca para aprendizagem de dados *multi-label*, que oferece um conjunto vasto de algoritmos de classificação, *ranking*, *thresholding* e redução de dimensionalidade, tal como algoritmos para aprendizagem de *labels* estruturadas hierarquicamente [30].

2.3 *Big data*

Atualmente vivemos num mundo que é impulsionado por dados. O volume de dados tem vindo a crescer de forma exponencial ao longo dos anos devido ao crescimento análogo do número de equipamentos que são capazes de produzir dados. Com a evolução da tecnologia, os equipamentos que outrora tinham apenas uma única finalidade, agora além possuírem um sistema de dados próprio, estão todos conectados numa rede em que se privilegia o desempenho, a escalabilidade e a baixa latência. Deste modo, surgiu a necessidade para a criação de novos sistemas que fossem capazes de aceder e analisar os dados de forma eficiente, É neste sentido que surge o termo *big data*.

2.3.1 Definição

Big data descreve uma estratégia holística de gestão de informação de modo a permitir a análise de grandes quantidades de informação, e, por sua vez, descobrir padrões e correlações de dados. Os termos *big data* e *big data analytics* são usados inseparavelmente. Isto reflete a opinião comum que ”big data”não se refere apenas ao problema da sobrecarga de informação, mas também às ferramentas de análise usadas para gerir o crescimento exponencial de dados e tornar esse crescimento em fontes de produtividade e informação útil. [**?**]

Outras definições salientam que para que os dados sejam classificados como *big data* devem possuir os três V: Volume, Variedade e Velocidade. [31]

- **Volume** - carateriza-se pela elevada quantidade de dados, as quais exigem elevado processamento;

- **Variedade** - Os dados são gerados em todos os tipos de formatos - dados estruturados, dados numéricos em bancos de dados tradicionais, e até documentos de texto não estruturados, *email*, vídeo, áudio, dados de cotações da bolsa e transações financeiras;

- **Velocidade** - Os dados fluem a uma velocidade sem precedentes e devem ser tratados em tempo útil. Tags de RFID, sensores, *smartphones* e contadores inteligentes impulsionaram a necessidade de lidar com elevadas quantidades de dados em tempo real.

No SAS (*Statistical Analysis System*) ainda se consideram duas variáveis adicionais que são:

- **Variabilidade** - o fluxo de dados não é constante ao longo do tempo. Como tal, é preciso ter em conta situações de picos de fluxo de dados sem por em causa os tempos de respostas;

- **Complexidade** - Dada a grande variedade de tipos de dados é necessário estabelecer correlações para tirar maior proveito dos mesmos [31].

Existem diversas técnicas de análise. A escolha depende do tipo de dados em causa, das tecnologias disponíveis, e das questões de pesquisa que se pretendem responder. As técnicas mais frequentes são: *Association rule learning*, *Data mining*, *Cluster analysis*, *Crowdsourcing*, *Machine Learning* e *Text Analytics*. Esta técnicas não são disjuntas, mas sim complementares:

- ***Association rule learning*** – estabelecimento de regras com intuito de encontrar relações entre variáveis. Um exemplo da aplicação desta técnica são os sistemas de recomendação que são empregues pela Netflix e a Amazon;
- ***Data mining*** - combinação de dados provenientes de análise estatística, inteligência artificial e sistemas de informação, com o intuito de descobrir padrões em grandes quantidades de dados;
- ***Cluster Analysis*** - tipo de *data mining* que divide grandes grupos de dados em grupos mais pequenos com maior grau de similaridade cujas características não são conhecidas à partida;
- ***Crowdsourcing*** - modelo de *sourcing* específico em que indivíduos ou organizações usam contribuições de utilizadores da *web* para obter serviços ou ideias;
- ***Machine learning*** - campo de estudo que fornece aos computadores a capacidade de aprender sem serem explicitamente programados;
- ***Text analytics*** - dado que grande parte dos dados gerados estão no formato textual, *text analytics* pode ser usada para extrair informação [32].

2.3.2 *Big Data Analytics*

Big Data Analytics pode ser organizada nas seguintes categorias: *Batch-oriented processing*, *Streaming processing*, OLTP e *Interactive ad-hoc queries and analytics*.

Batch-oriented processing

Parte do paradigma que um elevado volume de dados é armazenado primeiro e analisado posteriormente. Este paradigma é amplamente utilizado para tarefas como análise, ordenação e contagem de dados.

O *MapReduce*, da Apache Hadoop, é o modelo mais popular hoje em dia de *batch-oriented processing*. O termo *MapReduce* refere-se a duas tarefas distintas: Mapeamento e Redução. O primeiro passo consiste em dividir os dados em pequenos conjuntos e distribuí-los pelos nós da rede. O mapeamento consiste em desempenhar operações de filtragem e ordenação que são aplicadas simultaneamente em cada conjunto de dados gerados anteriormente. Cada dado individual passa a estar representado como um par chave-valor. A redução consiste em agregar os dados e obter o resultado final [32].

O HDFS – *Hadoop Distributed File System* – é o sistema de ficheiros que abrange todos os nós num cluster *Hadoop* para armazenamento de dados e permite ligar vários sistemas de ficheiros em muitos nós locais de modo a torna-lo num sistema de ficheiros global2 [33].

As *frameworks* Apache Pig e Apache Hive são usadas para expressar tarefas de *big Data* e análise através de *map reduce* [32] para grandes *clusters*. Porém, estas ferramentas só estão otimizadas para processamento de dados em disco, o que os torna lentos na exploração de dados e na aplicação de algoritmos de análise com passos múltiplos. É neste sentido que surge o Apache Spark, uma nova *framework* de *batch-oriented processing* que pode correr aplicações 40 vezes superior em relação às da Hadoop.

Spark

Apache Spark é uma plataforma de análise de dados distribuída em memória, com o intuito de melhorar o desempenho de jobs analíticos de batch, de jobs de machine learning iterativos, queries iterativas e processamento de grafos.

O Spark distingue-se dos demais pelo uso de *datasets* distribuídos resilientes (*Resilient Distributed Datasets* – RRD), os quais são ótimos para *pipepiling* de operações computacionais paralelas e são, por definição, imutáveis. Cada RDD ou é uma coleção armazenada num sistema de armazenamento externo (como um ficheiro em HFDS), ou um *dataset* derivado que foi criado pela aplicação de operadores noutro RDD. Deste modo, permite uma forma única de tolerância a faltas baseada em informação de linhagem [34].

A *framework* da Apache Spark é constituída por uma componente de SQL e processamento de dados estruturados (Shark), a componente Spark Streaming, uma biblioteca de *machine learning* (MLlib) e um motor que suporta a execução de grafos gerais (GraphX)

[34] Fornece também três formas de persistir os RDDs:

- Armazenamento em memória como se de um objeto Java desserializado se tratasse [35];
- Armazenamento em memória como dados serializados (espaço limitado); [35]
- Armazenamento em disco (um RDD é demasiado grande para manter em memória, e implica custos elevados de computação); [35]

A MLlib consiste num conjunto de algoritmos de machine learning comuns incluindo classificação, regressão, clustering, filtros colaborativos, redução de dimensionalidade, além das primitivas de otimização subjacentes. [10] Dentro dos algoritmos de classificação existem dois tipos: classificação binária (SVM's lineares, regressão logística, árvores de decisão e *naive bayes*) e classificação com classes múltiplas (árvores de decisão e *naive bayes*). Como já foi referido, a Apache Spark também tem uma componente de *streaming*. A ideia é tratar pequenas porções sequenciais de computação batch determinística em intervalos de tempo constantes. Os dados recebidos em cada intervalo são armazenados no *cluster*. Quando o intervalo termina, os dados recolhidos são reduzidos e agrupados através de operações paralelas determinísticas. Por fim, armazena os dados em RDDs *Resilient Distributed Datasets* [36].

É preciso salientar que o Apache Spark não requer Hadoop para operar, mas o seu paradigma paralelo requer um sistema de ficheiros partilhados para o uso ótimo de dados estáveis. A fonte estável pode ser S3, NFS ou, tipicamente HDFS. [37]

Diversas empresas e organizações utilizam Spark em suas aplicações. Entre elas, podemos destacar: Amazon, Baidu, eBay Inc. (agregação de logs de transações e análises), Yahoo!, Grupon, NASA JPL - Deep Space Network e Yahoo! [38].

Streaming processing

Tipo de processamento ideal para sistemas que necessitam de analisar os dados em tempo-real.

Os sistemas Storm, Spark Streaming (componente *streaming* do Apache Spark) e S4 (*Simple Scalable Streaming System*) são exemplos da aplicação desta metodologia. Os dois primeiros são abordados mais à frente. O terceiro encontra-se descontinuado desde 2014, como tal, não é abordado [39].

Storm

Apache Storm é uma plataforma *open-source* distribuída para o processamento de dados em tempo-real e define o seu fluxo de trabalho através de grafos acíclicos direcionados (*Directed Acyclic Grapsh* (DAG's)[40], denominados topologias. As topologias apenas terminam a sua execução caso o utilizador assim desejar ou se encontrar uma falha que não pode ser recuperada em execução.

Existem algumas similaridades com os sistemas Apache Hadoop. A grande diferença está no facto de executarem topologias em vez de MapReduce Jobs, os quais irão terminar a sua execução eventualmente. É caraterizado por ser escalável, tolerante a faltas, extensível e fácil de utilizar.

Storm pode ser utilizado para análises em tempo-real, *machine learning online*, computação contínua via RPC (*Remote Procedure Call*), ferramentas ETL (*Extract, Transform, Load*) entre outros [34].

Apache Storm segue o modelo de arquitetura *master-slave*, em que o nó *master* é denominado como *Nimbus* e os nós *slave* ou nós *workers* são denominados "supervisores". Uma topologia é representada como um conjunto de nós, os quais podem ser de dois tipos: o nó *master* e nó *worker*. O *Nimbus* que é semelhante ao *JobTracker* da Hadoop, e é responsável pela distribuição de código pelo *cluster* e pela atribuição de tarefas a cada máquina e monitorização de falhas. Os supervisores são responsáveis por atribuir tarefas às máquinas que os compõem, e iniciar e terminar processos, com base no que é atribuído pelo *Nimbus*. Cada nó executa um subconjunto de uma topologia. Por outras palavras, uma topologia consiste num conjunto de processos espalhados pelas diversas máquinas de um nó *worker* [41].

A coordenação e sincronização entre nós do *cluster* Storm ocorre através do Apache ZooKeeper. Além disso, Storm também tem uma interface web própria pela qual é possível efetuar pedidos, os quais são recebidos pelo *Nimbus* [34].

Diversas empresas e organizações utilizam o Apache Storm. Entre elas destacam-se: Twitter, Baidu, Spotify, Verisign, Yahoo! e Mercado Livre. [38]

OLTP (Online Transaction Processing)

Sistemas que usam bases de dados operacionais desenhados para lidar com um número elevado de transações que geralmente desempenham alterações nos dados. É usado sobretudo em sistemas RDBMS tradicionais. Contudo, estes sistemas não são apropriados para desempenhar tarefas com grandes volumes de dados. Deste modo, surgiu a necessidade de utilizar sistemas NoSQL (*Not Only SQL*) os quais são capazes de alcançar altos níveis de desempenho. Existem 4 tipos de sistemas NoSQL: orientado pares chave-valor, orientado a documentos, orientado a colunas e orientado a grafos. [32]

Interactive ad-hoc queries and analytics

Permite realizar *queries* a grandes fontes e *queries* a *interfaces* com latência muito baixa. Tem como pressuposto que as *queries* não devem demorar mais do que alguns segundos a executar mesmo no caso de Big Data, de modo a que os utilizadores possam reagir a alterações sempre que seja necessário. O sistema mais popular hoje em dia é o Drill.

2.4 Trabalho relacionado da Accenture

A equipa na qual fui inserido, é responsável pela componente OBO (*Operational Back-Office*, que é responsável por todo o sistema de *video tolling*, a qual é submetida constantemente a melhorias consoante as necessidades e problemas do cliente.

Tal como foi referido anteriormente, o tema *dynamic pricing* já foi abordado na tese de João Gomes. Como o sistema não cumpria os requisitos para poder ser integrado no projeto, não foi colocado em funcionamento. Como tal, nenhuma solução de *dynamic pricing* está a ser desenvolvida atualmente [4].

Hoje em dia, a *Illinois Tollway* é composta por cinco concessões: I-88, I-90, I-294, I-355, I-390. Cada uma apresenta características singulares as quais serão salientadas em capítulos posteriores. Existem duas formas de pagamento em funcionamento: I-PASS (uso de *tags* ou *transponders* ou ALPR) e em Dinheiro. Os preços aplicados variam de acordo a classe do veículo, modo de pagamento, e altura do dia.

Todos os utilizadores de veículos ligeiros (automóveis e motociclos), que possuam uma conta I-PASS têm um desconto de 50% no valor da portagem.

A tabela seguinte revela como é realizada a distinção de veículos e preços aplicados.

Tipo de Veículo	Classe de Veículo	Descrição	Variação de Preços
Veículos Ligeiros	A	Automóveis e Motociclos	Veículos com I-PASS têm um desconto de 50%
Veículos Pesados	B	Camião ou autocarro pequeno	Preços diurnos e noturnos
	C	Camião ou autocarro médio	Preços diurnos e noturnos
	D	Camião ou autocarro grande	Preços diurnos e noturnos

Tabela 2.3: Fatores envolvidos na aplicação de preços

2.5 Sumário

Após o levantamento do trabalho relacionado é possível iniciar a análise e desenho do serviço a desenvolver.

Capítulo 3

Análise

Como referi no capítulo anterior, irei basear-me nos princípios de *Congestion Pricing* descritos no livro "Traffic & Highway Engineering"de Nicholas J. Garber e Lester A. Hoel. Deste modo, todos os cálculos envolvidos no algoritmo de *dynamic pricing* são formulados no livro evidenciado. Na seção seguinte serão abordados: os requisitos, as fontes de dados e todos os conceitos necessários que envolvem a análise do estado de congestionamento de tráfego.

3.1 Requisitos

Os requisitos são uma peça fundamental em qualquer sistema. Os requisitos definem quais são os pressupostos que devem constar ou respeitar no projeto. Existem requisitos funcionais e não funcionais, sendo que os funcionais podem ser subdivididos em requisitos operacionais e de negócio.

3.1.1 Requisitos funcionais

1) **Requisitos Operacionais**

Código	Nome
RO-01	O sistema deve garantir que o flow rate previsto tenha um valor aceitável para a situação em questão.
RO-02	O sistema deve garantir que não é atribuído um preço de tarifa superior ao delimitado para cada LOS

Tabela 3.1: Requisitos Operacionais (RO)

2) **Requisitos de Negócio**

Código	Nome
RN-01	O sistema deve conseguir recolher as transações geradas nos pórticos das autoestradas através do OBO.
RN-02	O sistema deve conseguir recolher as informações do estado meteorológico atual.
RN-03	O sistema deve conseguir recolher o número de acidentes.
RN-04	O sistema deve conseguir criar as métricas do flow rate
RN-05	O sistema deve conseguir construir um modelo de aprendizagem
RN-06	O sistema deve conseguir prever o flow rate para os próximos 15 minutos
RN-07	O sistema deve quantificar o número de transações geradas.
RN-09	O sistema deve recolher as tarifas atuais em cada pórtico.
RN-10	O sistema deve calcular a densidade de tráfego para cada autoestrada que compõe a Illinois Tollway.
RN-11	O sistema deve calcular o LOS para cada pórtico tendo em conta a densidade de tráfego calculada anteriormente.
RN-12	O sistema deve conseguir atualizar os novos preços dinamicamente.
RN-13	O sistema deve apresentar uma interface gráfica que apresente análises estatísticas

Tabela 3.2: Requisitos de Negócio (RN)

3.1.2 Requisitos não funcionais

Código	Nome
Escalabilidade	
ESC-01	O sistema deve ser capaz de lidar com pelo menos 3.5 milhões de transações diárias.
ESC-02	O sistema deve ser capaz de lidar com picos de transações.
ESC-03	O sistema deve ser capaz de analisar um volume de transações anual
Desempenho	
DES-01	O sistema deve conseguir atualizar os preços a cada 15 min.
DES-02	O sistema deve apresentar uma precisão superior a 90\% na previsão do flow rate para cada plaza
Integridade	
INT-01	O sistema deve garantir que a perda de dados seja inferior a 1\%.
Disponibilidade	
DIS-01	O sistema deve estar operacional 24h/dia.
Usabilidade	
USA-01	A interface web deve respeitar os padrões de desenho do site da Illinois Tollway.
USA-02	Na interface web, as estatísticas das tarifas devem se apresentar de forma clara.
Recuperação	
REC-01	Em caso de falha a aplicação deve fazer *rollback* à operação que estava em execução.
Manutenção	
MAN-01	O sistema deve garantir mecanismos de manutenção do mesmo
Segurança	
SEG-01	O sistema deve garantir que os dados apenas são alterados por entidades com a devida autorização.
Resiliência	
RES-01	O sistema deve conseguir recuperar de uma falha e retomar a execução onde terminou.

Tabela 3.3: Requisitos Não Funcionais (RNF)

3.2 Algoritmo de análise do estado de congestionamento de tráfego

3.3 Fontes de Dados

Os dados fundamentais para análise do congestionamento de tráfego são as transações geradas nos pórticos das autoestradas. Como se pretende analisar dados em tempo-real, a base de dados do OBO será a fonte de dados do sistema. O projeto tem vários tipos de bases de dados consoante o nível de segurança e ambiente de programação:

1) **OBD** - Base de dados de Desenvolvimento
2) **OBQ** - Base de dados de Qualidade
3) **OBP** - Base de dados de Produção

Apenas me foi concedido o acesso a OBD e OBQ. O acesso a OBP seria a melhor opção dado que ira utilizar dados reais, portanto apresentam o melhor nível de confiança. Em OBQ existem simuladores que criam transações com um comportamento próximo da realidade. Como tal, embora não apresente o melhor nível de confiança, é possível apresentar resultados sólidos.

Será também necessário recolher as informações sobre as características físicas de uma autoestrada, e a tabela de tarifas base para cada pórtico, os quais serão armazenadas localmente.

Para aumentar o nível de confiança do modelo de aprendizagem é relevante ter em consideração eventos meteorológicos, eventos desportivos e feriados. O nível de tráfego é altamente influenciado pela ocorrência destes eventos. Deste modo, é estritamente necessário recolher estas informações de forma automática.

Em relação aos dados meteorológicos, estes serão recolhidos com auxílio da ***Weather Underground API***, na qual é possível recolher os eventos meteorológicos 24h/dia. É possível recolher 6 tipos de eventos: nevoeiro (*fog*), chuva (*rain*), neve (*snow*), vento (*hail*), trovoada (*thunder*) e tornados (*tornado*).

Em relação aos eventos desportivos e feriados, estes serão recolhidos com auxílio da ***Google Calendar API***.

Para uma melhor compreensão da estrutura da base de dados, para todas as tabelas serão evidenciados os campos respetivos incluindo também exemplos. É possível determinar o desempenho de uma estrada de formas diferentes consoante o seu tipo:

- Estradas com duas vias, uma para cada lado (*Two-lane Highways*)
- Estradas com múltiplas vias para cada lado (*Multi-lane Highways*)
- Autoestradas (*Freeways*)

Cada tipo de segmento de estrada tem as suas próprias características e formulas de cálculo do desempenho da mesma como a Taxa de Fluxo (*Flow Rate*), Velocidade livre de fluxo (*Free-Flow Speed*), densidade, etc. Dado que o tema de tese está inserido no projeto Illinois Tollway, o qual apenas possui auto-estradas, apenas me irei focar na descrição das características das *Freeways*. [42]

Caracterização das *Freeways*

Uma *freeway* é uma estrada com controlo total de acesso, como duas ou mais vias em cada direção com uso exclusivo de tráfego automóvel. Cruzamentos, interseções ou o uso direto de terras adjacentes não são permitidos para garantir o fluxo contínuo de veículos. O tráfego em direções opostas é separado por uma barreira física. Uma autoestrada é composta por três elementos: secções básicas de autoestrada , *weaving areas*, e vias de aceleração. Secções básicas são segmentos sem a influência dos outros dois elementos. A relação velocidade-fluxo-densidade depende do nível de tráfego corrente e das condições das estradas. As condições básicas de fluxo livre (base free-flow) incluem as seguintes características: [42]

- Vias com 12 pés de largura (3.6579 m)
- Margem total (margem esquerda e direita) com 6 ou mais pés.
- Não é permita a circulação de camiões, autocarros e auto-caravanas
- Autoestradas urbanas têm 5 vias em cada direção
- Interseções ocorrem no mínimo a cada 2 milhas de distância
- Grau de inclinação do terreno não pode exceder 2%
- Condutores são experientes em autoestradas.

Fatores de condicionamento do fluxo de tráfego

Uma medida importante utilizada para avaliar qualitativamente as condições operacionais do sistema de tráfego e, como estas condições são percepcionadas pelos condutores e passageiros, são os Níveis de Serviço (*Level of Service* – LOS). Esta medida está relacionada com as características físicas de cada *Freeway* e das várias situações que podem ocorrer com diferentes volumes de tráfego. As condições básicas de fluxo livre são os principais fatores que influenciam o nível de serviço atribuído a uma *Freeway*. Os fatores são os seguintes: [42]

- **Largura da via** – o fluxo de tráfego tende a ser restringido quando a largura das vias é inferior a 12 pés. Tal acontece porque a distância lateral entre veículos de vias adjacentes diminui, e por sua vez, os condutores tendem a diminuir a velocidade;
- **Margem** – quando as barreiras laterais das faixas se encontram muito próximas da via, os condutores tendem a afastarem-se desse lado da via. Por sua vez, a distância

lateral de veículos que circulam em vias adjacentes diminui, que se traduz numa redução da velocidade;

- **Veículos Equivalentes** – a presença de veículos longos como camiões e autocarros diminuem o fluxo máximo pelo seu tamanho, características operacionais e interação com outros veículos. Como a capacidade de uma autoestrada é expressa em carros por hora e por via (pc/h/ln), o número de veículos pesados em circulação deve ser convertido por um número equivalente de veículos ligeiros;
- **Inclinação** – o efeito da inclinação depende não só do grau de inclinação, mas também da extensão de estrada com essa mesma inclinação. As operações de tráfego são afetadas significativamente quanto a inclinação é igual ou superior a 3% em menos de um quarto de milha, e quando a inclinação é inferior a 3% em mais de 0.5 milhas;
- **Velocidade** – diretamente relacionado com a velocidade espacial média (média harmónica da velocidade dos veículos que passam num determinado ponto durante um intervalo de tempo);
- **Condutores-alvo** – refere-se às capacidades de condução dos condutores;
- **Comprimento das secções de autoestrada** – em áreas urbanas, as autoestradas costumam ser de pequenas. Como tal, a velocidade de fluxo livre (free-flow) tende a ser inferior do que em autoestradas de grande dimensão. [42]

Como já foi referido, as características de desempenho que descrevem os níveis de serviço (LOS) para uma secção básica de autoestrada são: a taxa de fluxo (Flow Rate, pc/h), velocidade média (*average passenger car speed*, mi/h) e a densidade que define o número de carros por milhas (pc/mi/ln). Existe uma relação entre as três características, a qual é definida mais adiante

Definição dos níveis de serviço

Para utilizar a formulação exata do livro, as formulas e componentes respetivas são referenciadas em inglês. Os níveis de serviço estão definidos de A a F. Cada nível tem uma taxa de fluxo, velocidade média e densidade associado [42].

- **Nível de serviço A** – As operações de fluxo livre são completamente independentes das capacidades de condução. Deste modo, o tráfego flui sem qualquer tipo de perturbação;
- **Nível de serviço B** – as operações de fluxo-livre são sustentáveis, mas as manobras de condução são ligeiramente restringidas. Como tal, podem existir algumas perturbações;
- **Nível de serviço C** – a velocidade média de tráfego é próxima da velocidade de fluxo-livre (*free-flow speed*), mas existem restrições de manobras em situações de mudança de via.

- **Nível de serviço D** – a velocidade média de tráfego diminui e a densidade aumenta mais rapidamente. A redução de liberdade nas manobras torna-se evidente e o espaço entre veículos da mesma via decresce.
- **Nível de serviço E** – a velocidade média de tráfego diminui significativamente. Manobras de mudança de via ou de entrada na autoestrada provocam perturbações no fluxo. Por esta altura, o aparecimento de filas extensas é comum.
- **Nível de serviço F** – quebra total do fluxo de trânsito. Dificuldades nas secções anteriores à autoestrada prevalecem durante largos períodos de tempo [42].
  O gráfico seguinte evidencia os intervalos de densidade, de velocidade média e flow rate para cada nível de serviço.

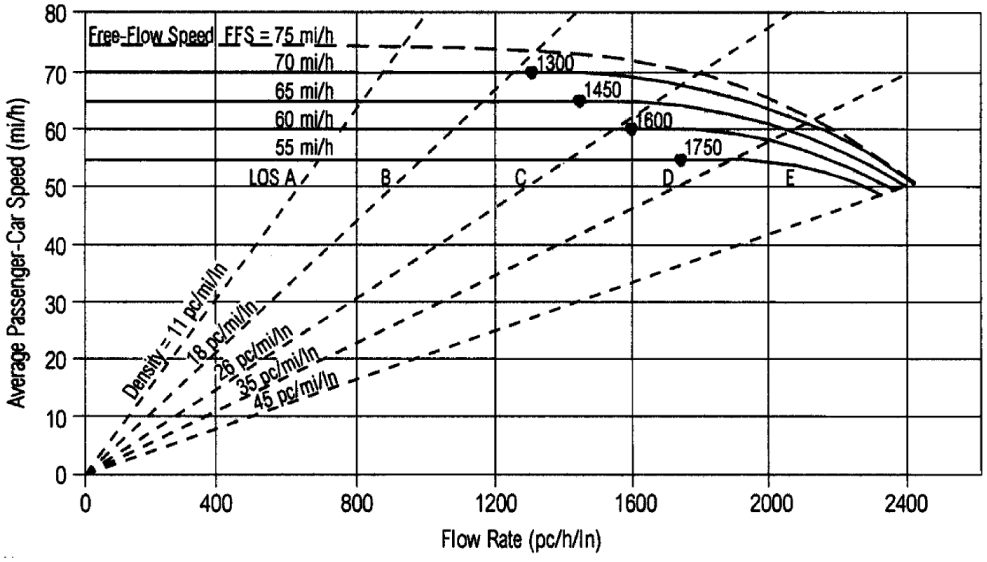

Figura 3.1: Curvas de *Speed-Flow* e LOS para segmentos básicos de autoestradas [42]

Cálculo do Nível de Serviço em seções básicas de auto-estradas (LOS)

Por motivos de simplificação e de ausência de informação no que toca ao relevo, todos os cálculos são referentes a seções básicas de auto-estradas. Para calcular o nível de serviço é necessário efetuar um conjunto de passos:

1) **Calcular o valor de *Free-Flow Speed* (FFS)**
   Consiste na velocidade livre de fluxo que se espera que os veículos circulem tendo em conta as características físicas da auto-estrada em questão.

$$FFS = BFFS - f_{LW} - f_{LC} - f_N - f_{ID} \tag{3.1}$$

FFS : *free-flow speed* (mi/h)
$BFFS$: base *free-flow speed* (mi/h) - por omissão para *freeways* é 75 mi/h (equivalente a 120 km/h)
f_{LW} : fator de ajustamento para a largura da via (lane width)
f_{LC} : fator de ajustamento para margem margem direita da faixa (lateral clearence)
f_N : fator de ajustamento para o número de vias
f_{ID} : fator de ajustamento para a densidade em zonas de entradas e saídas (*interchange density*)

* Os valores que cada fator podem tomar estão definidos em anexo. Alguns valores foram atribuídos por omissão, dado que não foi possível recolher informações sobre os mesmos.

- f_{LC} : Tabela A.2 Fator de ajustamento para a largura da margem direita da via (por omissão, 6 ft como margem direita)
- f_N : Tabela A.3 Fator de ajustamento para o nº de vias (diferente consoante a *freeway* em questão)
- f_{ID} : Tabela A.4 Fator de ajustamento para a densidade na zona de *interchanges* (por omissão, 1.0 por milha)
- f_{LW} : Tabela A.5 Fator de ajustamento para a largura da via (por omissão, 12 ft)

2) **Calcular do PHF (*peak hour factor*)**
Consiste numa medida para analisar o ponto mais alto de tráfego da última hora.

$$PHF = \frac{V}{4 * V_{15}} \tag{3.2}$$

V : Volume total num período de 1 hora
V_{15} : Volume máximo durante 15 min *
* O Volume de tráfego é medido a cada 15 min, logo 4 vezes por 1 hora. O período de 15 min em que se verifique maior tráfego corresponder ao V_{15}.

3) **Calcular o valor do *Flow Rate* (Fr)**
Consiste na taxa de fluxo de tráfego de veículos ligeiros por hora (*passenger cars per hour - pc/h*). Como circulam tanto veículos ligeiros como pesados, é necessário converter os veículos pesados em ligeiros.

$$v_p = \frac{V}{(PHF)(N)(f_p)(f_{HV})} \tag{3.3}$$

V : volume máximo por hora, numa direção (veh/h)
v_p : flow rate (pc/h)

PHF : fator de hora de ponta
N : número de vias numa só direção
f_p : fator da população condutora
f_{HV} : fator de ajustamento de veículos pesados

$$f_{HV} = \frac{1}{1 + P_T(E_T - 1) + P_R(E_R - 1)} \tag{3.4}$$

P_T e P_R : Percentagem de veículos pesados e percentagem de veículos recreativos em circulação respetivamente.
E_T e E_R : Número equivalente de veículos que usam o mesmo espaço que veículos pesados e veículos recreativos respetivamente.

* O fator da população condutora (f_p) foi definido por omissão como 1.0.

4) **Determinar o valor da velocidade média de tráfego (S)**
O cálculo depende do valor de FFS calculado anteriormente do seguinte modo: Se $v_p <= 30*FFS-3400$ então S = FFS. Equação da linha que liga os pontos dos *breakpoints* da curva da figura 3.1 com os valores 55, 60, 65, 70, e 75 mi/h. Se FFS $>$ 70 mi/h, então

$$S = FFS - \left[\left(FFS - \frac{160}{3}\right)\left(\frac{v_p + 30 * FFS - 3400}{30 * FFS - 1000}\right)^{2.6}\right] \tag{3.5}$$

Se FFS $<=$ 70 mi/h, então

$$S = FFS - \left[\frac{1}{9}(7 * FFS - 340)\left(\frac{v_p + 40 * FFS - 3400}{30 * FFS - 1700}\right)^{2.6}\right] \tag{3.6}$$

Nota: não é indicado no livro o significado de cada fórmula.

5) **Calcular o valor da Densidade (D)**

$$D = \frac{v_p}{S} \tag{3.7}$$

onde
D : Densidade (pc/mi/ln)
S : Velocidade média de tráfego (mi/h)
v_p : Flow Rate (pc/h/ln)

6) **Atribuição do LOS** associado ao valor de densidade calculado segundo a Tabela 3.4 (ver no anexo A).

	LOS				
Critério	A	B	C	D	E
FFS = 75 mi/h					
Densidade máxima (pc/mi/ln)	11	18	26	35	45
Velocidade máxima (mi/h)	75.0	74.8	70.6	62.2	53.3
Máximo v/c	0.34	0.56	0.76	0.90	1.00
***Flow rate* máximo**	820	1350	1830	2170	2400
FFS = 70 mi/h					
Densidade máxima (pc/mi/ln)	11	18	26	35	45
Velocidade máxima (mi/h)	70.0	70.0	68.2	61.5	53.3
Máximo v/c	0.33	0.53	0.74	0.90	1.00
***Flow rate* máximo**	720	1260	1770	2170	2400
FFS = 65 mi/h					
Densidade máxima (pc/mi/ln)	11	18	26	35	45
Velocidade máxima (mi/h)	65.0	65.0	64.6	59.7	52.2
Máximo v/c	0.30	0.50	0.71	0.89	1.00
***Flow rate* máximo**	710	1170	1680	2090	2350
FFS = 60 mi/h					
Densidade máxima (pc/mi/ln)	11	18	26	35	45
Velocidade máxima (mi/h)	30.0	60.0	60.0	59.7	52.2
Máximo v/c	0.29	0.47	0.68	0.88	1.00
***Flow rate* máximo**	660	1080	1560	2020	2300
FFS = 55 mi/h					
Densidade máxima (pc/mi/ln)	11	18	26	35	45
Velocidade máxima (mi/h)	55.0	55.0	55.0	54.7	50.0
Máximo v/c	0.27	0.44	0.64	0.85	1.00
***Flow rate* máximo**	600	990	1430	1910	2250

Tabela 3.4: Atribuição do LOS. Adaptado de (Traffic & Highway Engineering"de Nicholas J. Garber e Lester A. Hoel, 2009) [42]

3.4 Sumário

Após a realização da levantamento dos requisitos e fontes dados, estão reunidas as condições para iniciar o desenho do serviço. A análise do estado de congestionamento de tráfego apresentada foi fundamental para poder iniciar o desenvolvimento do algoritmo respetivo.

Capítulo 4

Desenho

Neste capítulo serão evidenciados a arquitetura de alto nível da solução, o modelo de domínio, os indicadores de desempenho, e os casos de uso do sistema.

4.1 Arquitetura de alto nível da solução

O sistema que se pretende desenvolver baseia-se numa arquitetura de 3 camadas: Ligação à Base de Dados, Lógica e Decisão e Apresentação. Uma possível abstração da sistema é o modelo de apresentação *multi-tier* na figura 4.1.

São apresentadas 3 camadas para diferenciar as várias funcionalidades presentes, logo, existe uma dedicada à interação com o utilizador e execução de serviços (*web container*), outra dedicada à implementação de toda a lógica (*ejb container*), e por fim uma dedicada ao mapeamento da informação recolhida do exterior em objetos java (*orm container*). O desenho desta arquitetura deve-se também à necessidade de criar um sistema que apresente uma estrutura semelhante ao projeto onde fui inserido na empresa. Uma outra alternativa passaria pela implementação de projeto java simples sem utilização de um servidor, que tornaria o desenvolvimento de software mais simples mas perderia algumas capacidades como a injeção de dependências e dificultaria o acesso e mapeamento da base de dados.

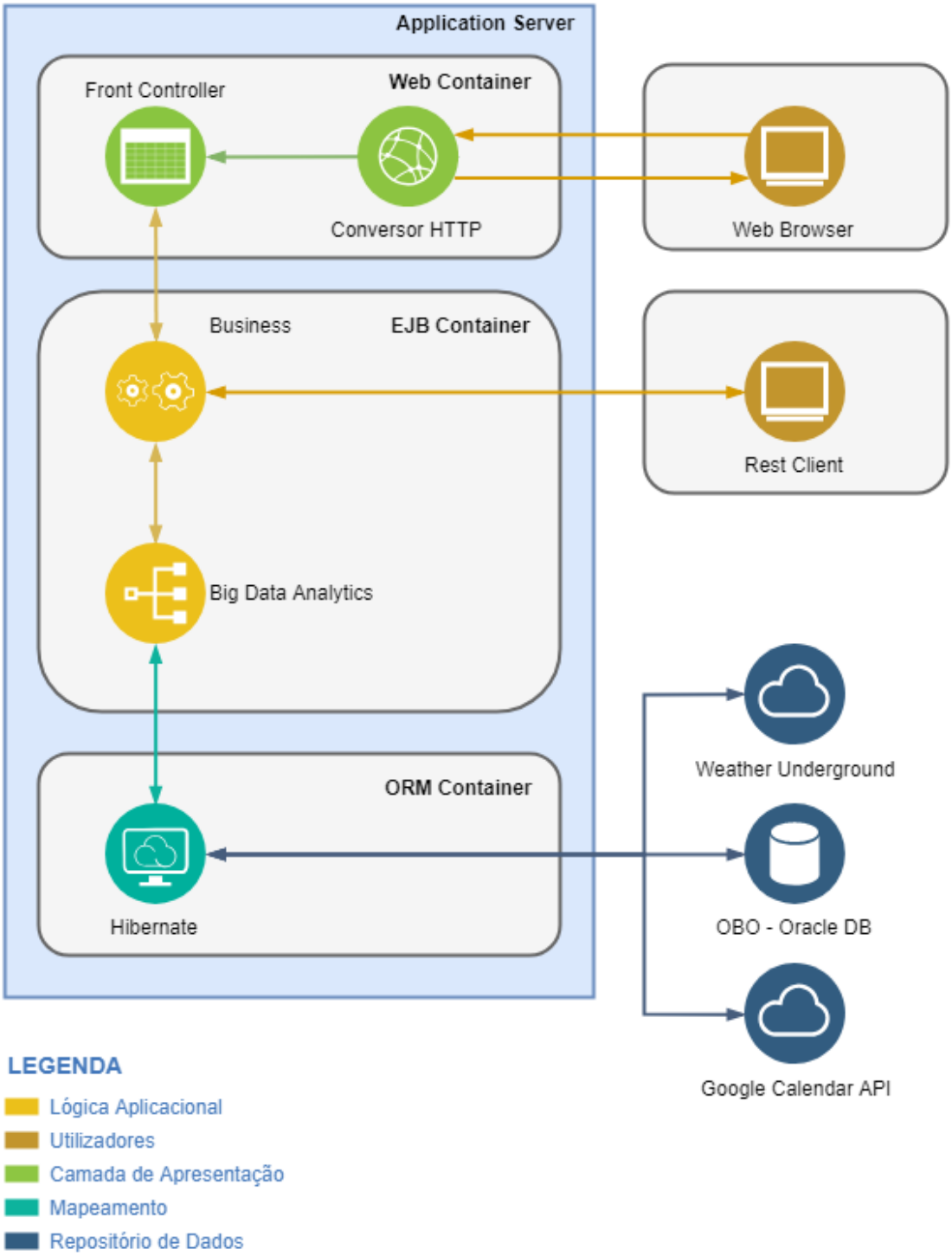

Figura 4.1: Arquitetura da solução

Catálogo de Elementos

- **Web Browser** - representa o lado do cliente. Interface HTML + CSS que interage com o cliente e que comunica com o resto do sistema através de HTTP;
- **Rest Client** - Aplicação que acede à lógica de negócio através de serviços REST;

- **Front Controller** - analisa o URL com a ajuda do ficheiro app.properties, cria o objeto responsável e passa o controlo para o objeto;
- **Business** - aplicação Java EE que conhece a lógica toda do sistema e fornece acesso remoto via serviços REST ao sistema;
- **Big Data Analytics** - filtragem dos dados mapeados em objetos java de acordo com os requisitos do sistema.
- **Hibernate** - mapeamento de dados de uma base dados relacional em objetos java;
- **Oracle DB** - repositório de dados relacional.

4.2 Modelo de Domínio

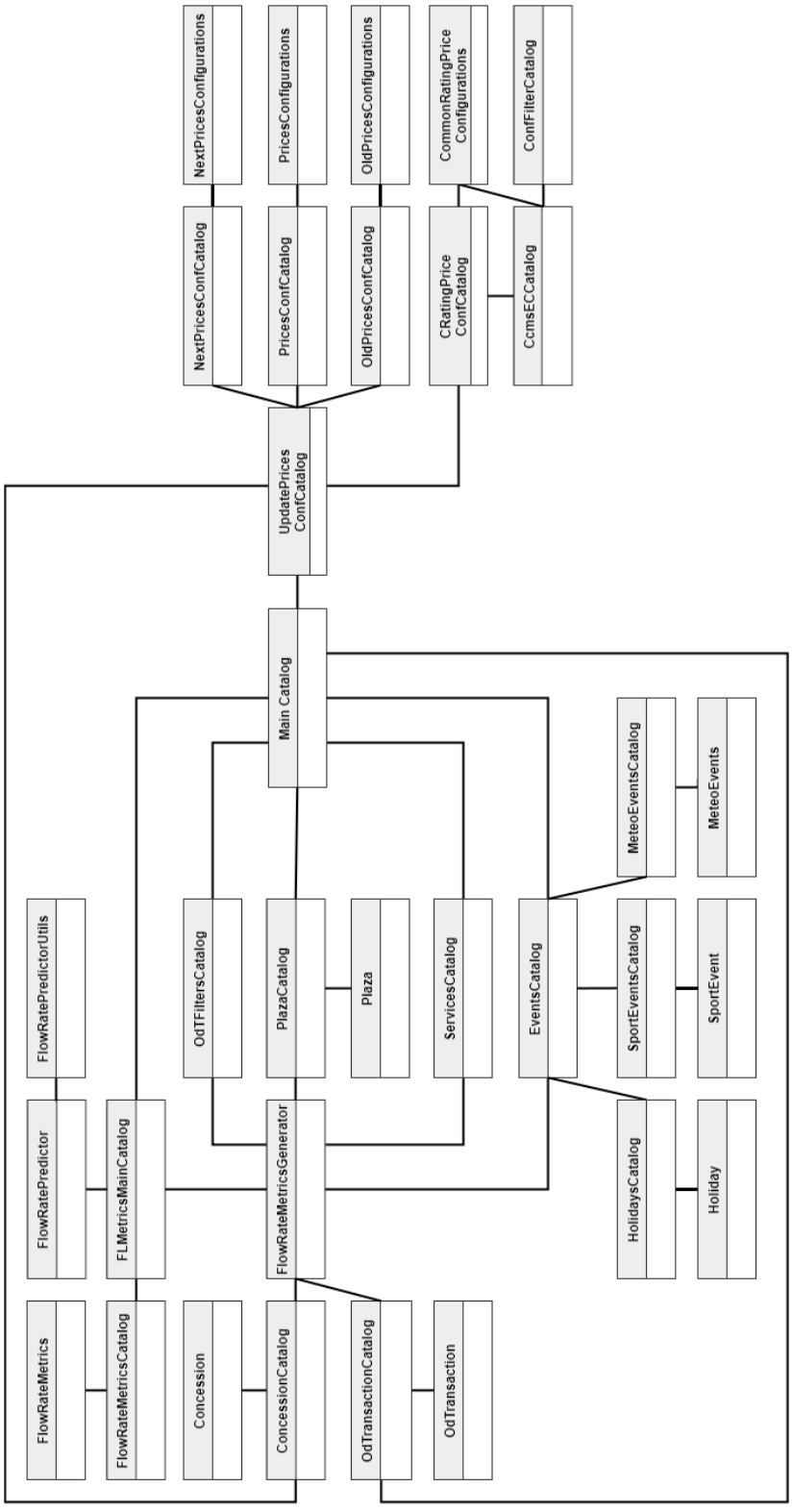

Figura 4.2: Modelo de domínio

O modelo de domínio representa a estrutura do projeto e respetivas interações. Estão representados catálogos e entidades. Os catálogos representam os *beans* de sessão e que contêm as interações e cálculos com as respetivas entidades que representam tabelas na base de dados. Pretende-se que haja uma clara diferenciação entre objetos tendo em conta à sua natureza e propósito, de modo a facilitar a interação entre os mesmos.

Catálogo de Elementos

- **MainCatalog** - responsável por toda lógica do sistema. (Session Bean);
- **ServiceCatalog** - representa todos os cálculos intermédios inerentes à identificação dos níveis de serviço (LOS) (Session Bean);
- **OdTransaction** - representam as transações gerados em cada plaza de cada concessão. (Entidade);
- **OdTrasactionCatalog** - representa o catálogo de OdTransactions, ou seja, conhece tudo sobre as OdTransactions. (Session Bean);
- **Concession** - representam as características físicas das concessões. (Entidade);
- **ConcessionCatalog** - representa o catálogo das concessões (Session Bean);
- **Plaza** - representam os pórticos das Concessões. (Entidade);
- **PlazaCatalog** - representa o catálogo de concessões (Session Bean);
- **PriceConfigurations** - representa as configurações preços atuais em cada plaza. (Entidade);
- **PricesConfCatalog** - representa o catálogo de configurações de preços atuais, incluindo todos os métodos necessários para aceder às configurações respetivas (Session Bean);
- **OldPriceConfigurations** - representa as configurações preços anteriormente aplicados em cada plaza, incluindo todos os métodos necessários para aceder às configurações respetivas (Entidade);
- **OldPricesConfCatalog** - representa o catálogo de configurações de preços anteriormente aplicados, ou seja, conhece tudo sobre as configurações de preços anteriores. (Entidade);
- **UpdatePricesConfCatalog** - representa o catálogo responsável pela atualização das configurações dos preços para cada plaza. (Session Bean);
- **CommonRatingPricesConfigurations** - representa as configurações base dos preços aplicados em cada plaza. (Entidade);
- **CRatingPricesConfCatalog** - representa o catálogo de configurações base de preços, incluindo todos os métodos necessários para aceder às configurações respetivas (Session Bean);
- **OdTFiltersCatalog** - representa o conjunto de operações de filtragem sobre as OdTransactions, das métricas de preços e previsão de preços. (Session Bean);
- **ConfFiltersCatalog** - representa o conjunto de operações de filtragem sobre as

configurações de preços. (Session Bean);

- **NextPricesConfigurationsCatalog** - representa o catálogo de configurações de preços que irão ser atribuídos no futuro, incluindo todos os métodos necessários para aceder às configurações respetivas. (Session Bean);
- **NextPricesConfigurations** - representa as configurações de preços que irão ser atribuídos no futuro. (Entidade);
- **FlowRateMetrics** - representa as métricas utilizadas no modelo de aprendizagem para prever o *flow rate* (Entidade);
- **FlowRatePredicator** - representa o catálogo que contém as operações de machine learning tanto para o treino do modelo de aprendizagem como também para a previsão do *flow rate* (Session Bean);
- **FlowRatePredictorUtils** - representa a classe com métodos auxiliares ao catálogo FlowRatePredicator.
- **FlowRateMetricsCatalog** - representa o catálogo de FlowRateMetrics, incluindo todos os métodos necessários para aceder às configurações respetivas. (Session Bean);
- **FlowRateMetricsGenerator** - representa o catálogo que contém todas as operações necessárias para gerar as FlowRateMetrics com base nas OdTransactions. (Session Bean);
- **FLMetricsMainCatalog** - representa o catálogo que contém todas as operações inerentes às FlowRateMetrics. (Session Bean);

  Nota: Também foram implementas outras classes auxiliares ao sistema mas que para motivos de simplificação do modelo de domínio não foram apresentadas.

4.3 *Key Performance Indicators* (KPIs)

Para que o sistema tenha sucesso é necessário que reflita no crescimento dos seguintes indicadores.

1) Percentagem do aumento da velocidade média;
2) Percentagem da redução da densidade de tráfego;
3) Percentagem do aumento do lucro;
4) Percentagem do aumento de adesão a *Freeways* menos congestionadas;
5) Aceitação dos condutores da nova metodologia de atribuição de preços.

4.4 Casos de Uso

Na secção seguinte serão evidenciados os diversos casos de uso internos envolvidos na execução do sistema. Como tal, não existe interação com atores. Os casos de uso identificam as várias operações envolvidas na execução do sistema. Inicialmente são identificados os casos de uso e posteriormente são devidamente detalhados.

4.4.1 Identificação dos Casos de uso

Código	Caso de Uso
UC1	Criar modelo de aprendizagem
UC2	Efetuar o cálculo da taxa de erro relativa ao modelo de aprendizagem
UC3	Efetuar a previsão do *flow rate*
UC4	Efetuar o cálculo do nível de serviço
UC5	Efetuar a atualização dos preços
UC6	Consultar a lista de plazas
UC7	Consultar a lista de concessões
UC8	Consultar o histórico de preços aplicados
UC9	Consultar o histórico do nível de serviço
UC10	Consultar o histórico de *flow rate metrics*.

Tabela 4.1: Identificação dos Casos de Uso

4.4.2 Documentação dos Casos de Uso

UC1 - Criar modelo de aprendizagem
Este caso de uso é responsável pelo processo da criação do modelo de aprendizagem. Como tal, engloba os passos de recolha de transações de um determinado período de tempo e, por sua vez, engloba a criação das *flow rate metrics* para cada período de 15 minutos que representam as métricas necessárias para avaliar cada período de tempo. Pretende-se criar um modelo credível e que tenha em consideração eventos meteorológicos, desportivos e feriados.

UC2 - Efetuar o cálculo da taxa de erro relativa ao modelo de aprendizagem
Este caso de uso é responsável por efetuar a análise de desempenho do modelo de aprendizagem através da verificação da taxa de erro inerente.

UC3 - Efetuar a previsão do *flow rate*
Este caso de uso é responsável por efetuar a previsão do *flow rate* dos próximos 15 minutos tendo por base as *flow rate metrics* referentes a esse instante. O valor do *flow rate* é a

classe que se pretende aprender no algoritmo de aprendizagem.

UC4 - Efetuar o cálculo do nível de serviço
Este caso de uso é responsável pela atribuição do nível de serviço de um determinado instante. Implica o cálculo do *Free Flow Speed*, *Flow Rate*, Velocidade Média e, por fim, Densidade.

UC5 - Efetuar a atualização dos preços
Este caso de uso é responsável pela atualização das configurações de preços de cada plaza. Como tal, implica não só atualizar as configurações atuais, as próximas que serão aplicadas, e o armazenamento das anteriores para poder consultar o histórico de configurações.

UC6 - Consultar a lista de plazas
Este caso de uso é responsável por efetuar uma consulta à lista plazas de todas as concessões da Illinois Tolway.

UC7 - Consultar a lista de concessões
Este caso de uso é responsável por efetuar uma consulta às características físicas inerentes a cada concessão da Illinois Tolway incluindo: largura da via, largura da margem, número de vias e número de entradas e saídas.

UC8 - Consultar o histórico de preços aplicados
Este caso de uso é o responsável por efetuar uma consulta ao histórico de preços aplicados calculados. Permite que o utilizador interaja com o sistema de modo a saber que o resultado das operações executadas, neste caso os LOS ao longo do tempo, para um dado Plaza.

UC9 - Consultar o histórico do nível de serviço
Este caso de uso é o responsável por efetuar uma consulta ao histórico de LOS calculados. Permite que o utilizador interaja com o sistema de modo a saber que o resultado das operações executadas, neste caso os LOS ao longo do tempo, para cada plaza.

UC10 - Consultar o histórico de flow rate metrics
Este caso de uso é responsável por efetuar uma consulta ao histórico de flow rate metrics calculados. Permite que o utilizador pesquise as *flow rate metrics* para cada plaza.

4.5 Sumário

Finalizada a análise e o desenho o próximo passo será a implementação do sistema. A fase de desenho é fundamental para que exista uma estrutura inicial pela qualquer projeto de software.

Capítulo 5

Implementação

Neste capítulo será evidenciado todo o processo envolvido na implementação, incluindo: o ambiente de desenvolvimento, as metodologias e ferramentas utilizadas no projeto, os processos envolvidos na recolha de dados para a aprendizagem, as etapas necessárias para a criação e treino do modelo de aprendizagem, a previsão propriamente dita dos novos preços a aplicar, e os mecanismos de visualização e execução do sistema.

5.1 Ambiente de Desenvolvimento

Para o desenvolvimento do sistema foi utilizado um computador portátil com as seguintes caraterísticas:

- Processador: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHZ
- RAM: 16 GB
- Sistema Operativo: Windows 10 Enterprise Edition 64 bits

Como IDE de programação foi utilizado o Red Hat JBoss Developer Studio, o IDE da Red Hat. Todo o códido desenvolvido foi implementado em Java, não só por ser a linguagem com a qual tenho mais à vontade, como também ser a linguagem utilizada no projeto de *tolling* onde estou inserido. Além disso, a linguagem Java tem uma vasta biblioteca de funções.

5.2 Ferramentas e Metodologias

A abordagem de *Congestion Pricing* ideal é a abordagem dinâmica, dado que tem em consideração mais fatores que podem influenciar o normal fluxo de tráfego. Como tal, será possível aplicar preços das tarifas mais adequados.

A escolha de software para um projeto é inteiramente dependente da natureza do problema em questão. No contexto do projeto Illinois Tollway, a latência é um fator importante, dado que são geradas transações na ordem dos milissegundos. Embora exista um

atraso entre a geração e o armazenamento das transações na ordem de alguns minutos, o armazenamento ocorre de forma contínua, ou seja, ocorre via *streaming*. A persistência de dados ocorre via Hibernate sob JBoss, um servidor aplicacional. A versão utilizada do JBoss foi a 6.4.0, que é a versão utilizada no projeto corrente.

O sistema implementado representa uma aplicação *Java EE Enterprise* e foi organizado em 3 camadas que são encapsuladas num único projeto:

1) dynamicPricing-ejb - camada que representa o módulo ejb, e que reúne toda a lógica aplicacional do sistema, portanto, consiste em todas as classes java para *enterprise beans*, e opcionalmente um *deployment descriptor*. O módulo EJB é empacotado num ficheiro JAR (*Java Archive*), que para além de conter todas as packages e classes java, contém todas as dependências deste módulo.
2) ***dynamicPricing-web*** - camada que representa o módulo web, e que consiste na interface web e chamada de serviços do módulo ejb, portanto, contém Servlets, ficheiros html, xhtml, imagens, etc. Além disso contém um *web application deployment descriptor*. O módulo web é empacotado num ficheiro WAR (Web Archive) contendo todos os ficheiros inerentes.
3) ***dynamicPricing-ear*** - camada que representa o módulo ear, e que empacota os módulo ejb e web, num único JAR, o qual é depois "*deployed*"no servidor.

As componentes fundamentais numa aplicação Java EE são os *Enterprise Beans*, que representam componentes java *server-side* e encapsulam a lógica de negócio da aplicação. Para esta aplicação foram utilizados *Stateless Session Beans*. Um *Session Bean* pode ser invocado pelo cliente seja através de uma via local, remota, ou *web services*. Deste modo, o cliente quando pretende interagir com a aplicação necessita de invocar os métodos respetivos a um determinado *bean* de sessão. Como não se pretende guardar o estado das variáveis foram implementados beans de sessão *Stateless* (sem estado). [43]

A metodologia de desenvolvimento utilizada foi a *Waterfall*, como tal, a realização do projeto teve a seguinte sequência: Análise, Desenho, Implementação e Testes.

O passo a seguir foi necessariamente decidir quais seriam as ferramentas a utilizar tendo em conta o problema em questão.

5.2.1 Ferramenta de recolha e tratamento de dados

Os sistemas de análise de dados *streaming* precisam de processar e gerir dados stream de forma rápida e eficiente, portanto é fundamental usar um modelo que seja capaz de escalar com a quantidade de dados.

Numa primeira abordagem fazia sentido a utilização do Apache Storm, dado que, este é puramente dedicado a *streaming* e apresenta uma latência inferior a 1 seg. Contudo, após detalhar as etapas envolvidas no processamento de transações, verificou-se que apenas seria viável analisar as transações geradas a cada 15 min. Isto significa que em média,

iriam ser analisadas mais de 36000 transações. Portanto, estamos perante uma análise de *batch processing*. Deste modo, o uso da Apache Spark será uma mais valia para o projeto de tese. Para além de ser amplamente utilizado, apresenta uma biblioteca de ML própria (MLlib).

5.2.2 Ferramenta para aplicação do algoritmo de ML

O passo seguinte é escolher a biblioteca de ML. Foram evidenciadas as bibliotecas mais comuns utilizadas hoje em dia, e todas têm algoritmos de classificação, como tal, todas seriam candidatas a implementar no projeto. A framework Apache SAMOA destaca-se dos de mais dado que foi desenvolvida especificamente para resolver problemas de *data mining* para *streams*. Motores de processamento de *streaming* distribuído (Distributed stream processing engines - DSPEs), como é o caso do Storm, combinado com algoritmos de *streaming* eficientes com processamento distribuído e escalável, aumenta significativamente o desempenho dos sistemas.

Mais uma vez foi necessário mudar de perspetiva. Tendo em conta que estamos perante um problema de *batch processing*, a Apache SAMOA já não é apropriada. Como já foi referido, a *framework* Apache Spark tem a sua própria biblioteca de ML, logo, o algoritmo a utilizar será, naturalmente, um dos algoritmos da biblioteca.

Além disso, estamos perante um problema de classificação com mais do que duas classes. Como tal, os algoritmos que compõem e cumprem os requisitos são o ***Naive Bayes***, ***Decision Tree*** e ***Random Forest***.

Para poder escolher o algoritmo ideal de forma fundamentada é necessário efetuar testes. Decidi avaliar os algoritmos pela taxa de erro e tempo de execução. A *framework* tem disponível exemplos de implementação dos algoritmos que se baseiam num ficheiro do tipo LIBSVM. Deste modo, executei 3 ciclos de 200 testes para cada algoritmo. Os resultados estão nas tabelas seguintes.

Taxa média de erro (%)		
Decision Tree	***Random Forest***	***Naive Bayes***
4.623	1.621	1.591
4.861	1.216	1.358
4.216	1.363	1.419

Tabela 5.1: Comparação de algoritmos por taxa média de erro de previsão

Analisando os resultados é possível constatar que a *Decision Tree* apresentou os piores resultados em termos de taxa de erro, e o *RandomForest* apresentou os piores resultados em termos de tempo de execução. O *Naive Bayes* foi o algoritmo que apresentou melhores resultados em ambos os testes. Embora o *RandomForest* apresente uma taxa de erro inferior, quando comparado com o *Naive Bayes*, a discrepância é pouco significativa. Por outro lado, em termos de tempo de execução, o *Naive Bayes* é claramente melhor. Dado

Tempo médio de execução (s)		
Decision Tree	***Random Forest***	***Naive Bayes***
0.890	0.975	0.600
0.746	0.803	0.628
0.681	0.764	0.572

Tabela 5.2: Comparação de algoritmos por tempo médio de erro de previsão

que se pretende atualizar os preços das tarifas nos pórticos tendo como base mais de 30 mil transações em intervalos constantes de 15 minutos, é fulcral executar o algoritmo o mais rapidamente possível. Assim, o algoritmo ideal para o problema é o *Naive Bayes*. De seguida irei entrar em detalhe na definição e funcionamento do algoritmo.

Algoritmo *Naive Bayes*

O algoritmo *Naive Bayes* é um dos algoritmos de aprendizagem mais eficientes para problemas de *machine learning* e *data mining*. Em *machine learning*, os classificadores *Naive Bayes*, como já foi referido, fazem parte do grupo de algoritmos probabilísticos e que partem do pressuposto de uma grande independência entre as características, atribuindo uma probabilidade a cada característica consoante o que se pretende prever. Tipicamente é utilizado para a classificação de documentos.

Um classificador é uma função que atribui uma classe a um determinado exemplo com base num modelo de aprendizagem previamente criado. Pela perspetiva probabilística, e tendo em conta a Regra de *Bayes*, a probabilidade de um exemplo $E = (x_1, x_2, ..., x_n)$ pertencer a uma classe c é

$$p(c|E) = \frac{p(E|c) * p(c)}{p(E)} \tag{5.1}$$

Assumindo que todos os atributos são independentes tendo em conta ao valor da classe, então

$$p(E|c) = p(x_1, x_2, ..., x_n|c) = \prod_{i=1}^{n} p(x_i|c) \tag{5.2}$$

[44]

O Spark, como já foi referido, suporta o algoritmo *Naive Bayes* na sua biblioteca de *machine learning*, conseguindo numa única passagem pelos dados de treino, calcular a distribuição de probabilidade condicional para cada característica de uma classe. Por sua vez, permite aplicar o teorema de *Bayes* para prever a classe de um dado exemplo. A spark.mllib suporta dois tipos de distribuições *Naive Bayes*: *Multinomial* e *Bernoulli*. A grande diferença entre as duas distribuições é evidencia nas seguintes fórmulas.

$$Multinomial - P(d|c) = P(< t_1, ..., t_k, ..., t_{nd} > |c) \tag{5.3}$$

$$Bernoulli - P(d|c) = P(< e_1, ..., e_k, ..., e_M > |c) \quad (5.4)$$

Onde $< t_1, ..., t_n d >$ é a sequência de termos na ordem de *d* (menos os termos que foram excluídos do vocabulário) e $< e_1, ..., e_k, ..., e_M >$ é um vector binário de dimensão *M* que indica para cada termo se ocorre ou não em *d*. Por exemplo, se pretendemos classificar países com base num conjunto de termos, então no caso da distribuição multinomial cada termo corresponde ao valor da característica (continente = Europa, capital = Lisboa), e no caso da distribuição de *bernoulli* os termos correspondem à existência ou não de uma característica (europa = 1, lisboa = 1). Permite também aplicar um fator de alisamento utilizando o parâmetro λ (por defeito, 1.0).

A forma como a biblioteca foi utilizada será evidenciada com mais detalhe nos próximos capítulos.

5.2.3 Ferramentas para realização de testes

Os *enterprise beans* são executados sob um EJB container, como tal, para poder realizar testes à aplicação é necessário poder simular o comportamento de um container. É neste sentido que surge a necessidade de utilizar a *framework* de teste **Arquillian** integrada com **JUnit** para executar e interagir com os *beans* de sessão. O facto de ser integrado com a *framework* JUnit permite escrever testes Java de forma clara e compreensiva para verificar o comportamento da aplicação. [45]

5.3 Recolha de dados para aprendizagem

A recolha de dados de aprendizagem ocorre em duas fases. A primeira consiste na análise e armazenamento de eventos (feriados, eventos desportivos e eventos meteorológicos), e a segunda consiste na análise das transações do último ano.

5.3.1 Análise e armazenamento eventos do último ano

Feriados

A *Google Calendar API* devolve, entre outras informações, o nome e data dos feriados. Cada feriado irá ser convertido num objeto *Holiday*. Todos os objetos *Holiday* são adicionados a uma lista e posteriormente armazenados na tabela HOLIDAY. A Tabela 5.1 evidencia os campos da tabela HOLIDAY.

Eventos desportivos

A *Google Calendar API* também devolve, entre outras informações, o nome e data de realização dos eventos desportivos. Cada evento irá ser convertido num objeto *SportEvent*.

Campo		Tipo de Dados	Obrigatório	Descrição	Exemplo	Obs.
1	*Id*	Numérico	Sim	Identificador do Feriado (BD)	1	PK
2	*Designation*	Varchar2	Sim	Designação do Feriado	Thanksgiving Day	-
3	*Holiday Date*	Timestamp	Sim	Data do Feriado	17.11.23 00:00:00,000 000000	-

Tabela 5.3: Tabela HOLIDAY e seus atributos

Todos os objetos *SportEvent* são adicionados a uma lista e posteriormente armazenados na tabela SPORT_EVENT. A Tabela 5.2 evidencia os campos da tabela SPORT_EVENT. Apenas são considerados os eventos desportivos das modalidades de grande relevância em Chicago e nas ligas mais importantes, as quais são: Basquetebol, Hóquei no Gelo, Basebol e Futebol Americano. Para cada modalidade existe um clube respetivamente: Bulls, Blackhwaks, Cubs e Bears.

Campo		Tipo de Dados	Obrigatório	Descrição	Exemplo	Obs.
1	*Id*	Numérico	Sim	Identificador do Evento Desportivo (BD)	1	PK
2	*Designation*	Varchar2	Sim	Designação do Evento Desportivo	Bears (10) @ Vikings (38)	-
3	*Sport Event Date*	Timestamp	Sim	Data do Evento Desportivo	17.01.01 18:00:00,000000000	-

Tabela 5.4: Tabela SPORT_EVENT e seus atributos

Eventos Meteorológicos

A *Weather Underground API* devolve, entre outras informações, 6 tipos de eventos meteorológicos como já foi referido, incluindo também o dia e hora em que ocorreram. A existência de eventos é evidenciada de forma binária: valor '1' caso ocorra, ou '0' caso contrário. A API foi utilizada no modo gratuito, o qual possui limitações como o número de pedidos. Apenas se podem efetuar 10 pedidos por minuto e 500 por dia. A API fornece um conjunto de operações de pesquisa de eventos meteorológicos sob a forma de um URL. A API permite verificar o histórico de eventos meteorológicos por dia, e o resultado é devolvido na forma de resposta JSON ou XML. O URL seguinte evidencia a estrutura exemplo de um pedido.

$$http://api.wunderground.com/api/key/history\_20170101/q/US/Chicago.json$$

Cada evento irá ser convertido para um objeto *MeteoEvents*. Todos os objetos *MeteoEvents* são adicionados a uma lista e posteriormente armazenados na tabela METEO_EVENTS.

A Tabela 5.3 evidencia os campos da tabela METEO_EVENTS. Desta forma, quando se pretender recuperar os eventos meteorológicos de um determinado momento, basta identificar os eventos com valor '1'.

Campo		Tipo de Dados	Obrigatório	Descrição	Exemplo	Obs.
1	*Id*	Numérico	Sim	Identificador do Evento Meteo. (BD)	614	PK
2	*Fog*	Numérico	Sim	Nevoeiro	0	-
3	*Rain*	Numérico	Sim	Chuva	0	-
4	*Snow*	Numérico	Sim	Neve	1	-
5	*Hail*	Numérico	Sim	Vento	0	-
6	*Thunder*	Numérico	Sim	Trovoada	0	-
7	*Tornado*	Numérico	Sim	Tornado	0	-
8	*Meteo Events Date*	Timestamp	Sim	Data do Evento Meteo.	18.04.16 01:00:12,000000000	-

Tabela 5.5: Tabela METEO_EVENTS e seus atributos

5.3.2 Análise das transações geradas no último ano

Como já foi referido, os dados fundamentais para o modelo de aprendizagem são as transações que são geradas nos pórticos dos plazas. Uma transação de negócio é um agregado de eventos, por parte de um único veículo registados junto dos pórticos. Pretende-se criar um modelo com elevado nível de confiança. Como tal, é necessário recolher as transações do último ano. Os preços serão atualizados em intervalos de tempo de 15 minutos, logo é relevante calcular o *flow rate* do último ano em intervalos constantes de 15 minutos. Portanto, a lista de transações desse período serão utilizadas para calcular o *flow rate*.

Como a recolha de transações ocorreu via VPN, o tempo de resposta era elevado e, consequentemente, o tratamento das transações e cálculo do *flow rate* acarretou muito tempo. Se as transações são analisadas em períodos de 15 minutos, então um dia tem 96 iterações, e por sua vez, um ano tem 35040 iterações. Analisar em tempo útil todas as transações da base de dados tornou-se inviável. Relembro que são geradas em média 3.5 milhões de transações por ano. Como tal, para fins de teste do sistema, apenas se considerou executar o sistema para um único plaza como prova de conceito. Depois de analisar o histórico de transações da tabela OD_TRANSACTION, verificou-se que o plaza com o atributo EXIT_POINT = 3003 apresentava dados suficientemente consistentes. Sendo assim, uma iteração, em média, demora cerca de 20 segundos.

A tabela que armazena as transações é a OD_TRANSACTION. A Tabela 5.4 evidencia os campos da tabela OD_TRANSACTION necessários para a execução do sistema.

Os preços das tarifas aplicadas nos pórticos que servirão de base para a aplicação do algoritmo encontram-se na tabela CCMS_ENTITY _CONFIGURATION. Esta tabela

	Campo	Tipo de Dados	Obrigatório	Descrição	Exemplo	Obs.
1	*Id*	Numérico	Sim	Identificador da Transação	1	PK
2	*Passage Type Id*	Numérico	Sim	Tipo de passagem	7	FK
3	*Concession Id*	Numérico	Sim	Referência para a concessão onde a transação ocorreu	301	-
4	*Payment Type*	Numérico	Sim	Tipo de pagamento	7	FK
5	*Exit Point*	Varchar2	Sim	Portagem de saída	"3001"	-
6	*Tariff Class*	Numérico	Sim	Classe do veículo indicado no seu dispositivo eletrónico	1	-
7	*Exit Point Direction*	Numérico	Sim	Direção da portagem de saída	1	-

Tabela 5.6: Tabela OD_TRANSACTION e seus atributos

já existe no projeto onde fui inserido. Como tal, apenas foi necessário replicar a sua estrutura. A Tabela 5.5 evidencia a estrutura da mesma.

	Campo	Tipo de Dados	Obrigatório	Descrição	Exemplo	Obs.
1	*Id*	Numérico	Sim	Identificador da Ccms Entity Configuration	13411	PK
2	*External Id*	Numérico	Sim	Identificador Externo	3031	-
3	*Conf Group Id*	Numérico	Sim	Identificador do Grupo de Configurações	109	FK
4	*Configuration Value*	Clob	Sim	Valor da Configuração	(*)	-
5	*Effective Start Date*	Date	Sim	Data de Início da Configuração	15.08.13	-
6	*Effective End Date*	Date	Sim	Data de Fim da Configuração	17.01.01	-
7	*Db Insert Date*	Date	Sim	Data de Inserção da Configuração	16.12.16	-

Tabela 5.7: Tabela CCMS_ENTITY_CONFIGURATION e seus atributos

Esta tabela contém múltiplas configurações do projeto. Como apenas são necessárias as configurações inerentes às tarifas, apenas estas foram acedidas. A informação requerida foi armazenada como uma string XML. Assim, foi necessário efetuar previamente o mapeamento da mesma em objetos Java. De seguida é evidenciado um exemplo de uma configuração que corresponde ao valor do campo *Configuration Value*.

Dado que estes dados são estáticos, apenas foi efetuado um acesso à tabela evidenciada e, os dados foram replicados para um tabela local denominada C_RATING _PRICES_CONF, como é evidenciado na Tabela 5.6. Cada campo xml da imagem anterior é representado numa coluna da tabela. Esta tabela contém todas informações necessárias referentes às configurações de preços. Deste modo, sempre que se pretenda atualizar os preços, basta aceder à mesma para identificar o preço base de uma determinada tarifa.

```xml
<config>
    <company companyName="ISTHA">
        <concession concessionShortName="I-90">
            <plaza plazaId="3031">
                <ratingPrices>
                    <rateTier>4</rateTier>
                    <cronExpressionText>* * 22-5 * * ? *</cronExpressionText>
                    <cronExpression-Locale>America/Chicago</cronExpressionLocale>
                    <rateType>AVI</rateType>
                    <price>170</price>
                    <overwriteSourcePrice>-</overwriteSourcePrice>
                    <effectiveStartDate>2015-08-13 05:00:00</effectiveStartDate>
                    <effectiveEndDate>2017-01-01 06:00:00</effectiveEndDate>
                    <configComments/>
                </ratingPrices>
            </plaza>
        </concession>
    </company>
</config>
```

Figura 5.1: Exemplo de uma configuração de preços

Campo		Tipo de Dados	Obrigatório	Descrição	Exemplo	Obs.
1	*CRPC Id*	Numérico	Sim	Identificador da Configuração da Tarifa	2	PK
2	*Concession Short Name*	Varchar2	Sim	Nome Simplificado da Concessão	I-355	-
3	*Concession Id*	Numérico	Sim	Identificador da Concessão	3062	-
4	*Plaza Id*	Varchar2	Sim	Identificador do Plaza	3190	-
5	*Lane Id*	Varchar2	Sim	Identificador da Via	39066	-
6	*Rate Tier*	Varchar2	Sim	Identificador da Classe de Veículo	3	-
7	*Cron Expression Text*	Varchar2	Sim	Cron Expression para representar a altura do dia	* * 22-5 * * ? *	-
8	*Rate Type*	Varchar2	Sim	Identificador do Tipo de Pagamento	AVI	-
9	*Price*	Numérico	Sim	Preço da Configuração	105	-
10	*Overwrite Source Price*	Varchar2	Sim	-	-	-
11	*Effective Start Date*	Timestamp	Sim	Data de Início da Configuração	15.01.01 06:00:00,0000000 00	-
12	*Effective End Date*	Timestamp	Não	Data de Fim da Configuração	17.01.01 06:00:00,0000000 00	-

Tabela 5.8: Tabela C_RATING _PRICES_CONF e seus atributos.

Em relação às características físicas das concessões, estas foram guardadas também numa tabela local denominada CONCESSION e com os atributos evidenciados na Tabela 5.7. (Atribuiu-se o nome *Concession* à tabela para manter a coerência de termos com o projeto. Os dados referentes às concessões serão relevantes para o cálculo do *free flow speed*.

	Campo	Tipo de Dados	Obrigatório	Descrição	Exemplo	Obs.
1	*Id*	Numérico	Sim	Identificador do Plaza (BD)	1	PK
2	*Concession Id*	Numérico	Sim	Identificador da Concessão	300	-
3	*Number of Lanes*	Numérico	Sim	Número de vias num único sentido	4	-
4	*Margin Right*	Numérico	Sim	Margem direita de uma faixa num único sentido	6 ft	-
5	*Interchange Density*	Float	Sim	Densidade de entradas e saídas numa concessão	1	-
6	*Concession Short Name*	Varchar2	Sim	Nome abreviado da concessão	I-88	-
7	*Lane Width*	Numérico	Sim	Largura de uma via	12 ft	-

Tabela 5.9: Tabela CONCESSION e seus atributos.

Para fins de reduzir o número de acesso à tabela OD_TRANSACTION, foi também criada uma tabela denominada PLAZA, que como o nome indica, contém todos os plazas que constituem a Illinois Tolway, como é evidenciado na Tabela 5.8.

	Campo	Tipo de Dados	Obrigatório	Descrição	Exemplo	Obs.
1	*Id*	Numérico	Sim	Identificador do Plaza (BD)	1	PK
2	*Plaza Id*	Varchar2	Sim	Identificador do Plaza	3056	-
3	*Concession Id*	Numérico	Sim	Identificador da Concessão	300	

Tabela 5.10: Tabela PLAZA e seus atributos.

OS atributos *Plaza Id* e *Concession Id* estão presentes em todas as tabelas. Deste modo, é possível recolher toda informação referente a um plaza sabendo apenas o seu Id.

Após a criação de todas as tabelas evidenciadas, o próximo passo é a extração de toda a informação necessária.

Gerador de Transações

Para fins de teste, inicialmente foi criado um gerador de transações próprio. Para ter algum nível de fiabilidade baseei-me num relatório da Illinois Tolway onde era descriminado o volume mensal de tráfego, especificando também o volume de veículos de passageiros e comerciais. Como o relatório é de Novembro de 2017, não existe informação do ano inteiro, logo, utilizei os valores de 2016. [46]
O cálculo do *flow rate* será realizado a cada 15 minutos. Deste modo, o volume mensal

será repartido nesses mesmos intervalos. Como não existe mais informações descriminadas para o ano inteiro, foi necessário atribuir fatores de aleatoriedade que são evidenciados de seguida:

- **Dia da Semana** - é atribuído uma percentagem de volume de tráfego consoante o dia da semana, da seguinte forma:

  1) Segunda - 1.25
  2) Terça - 1.15
  3) Quarta - 1.05
  4) Quinta - 1.0
  5) Sexta - 0.95
  6) Sábado - 0.85
  7) Domingo - 0.75

- **Classe de veículo** - só existe informação do volume de tráfego referente aos veículos de passageiros e comerciais. Como é de esperar que o volume de veículos ligeiros e motociclos referentes à classe 1 seja muito maior que as restantes, assumi que o volume de transações referente aos veículos de passageiros corresponde ao volume de veículos da classe 1. Para o volume de transações de veículos comerciais dividi o volume pelas restantes através da geração de um número aleatório entre 0 a 100, da seguinte forma:

  1) Entre 0 a 34, atribuiu-se a transação à classe 2;
  2) Entre 35 a 69, atribuiu-se a transação à classe 3;
  3) Entre 70 a 100, atribuiu-se a transação à classe 4;

- **Por plaza** - tendo a lista plaza para a concessão já calculada anteriormente, atribuiu-se um plaza com base no nº de elementos da lista.
- **Por direção de saída** - um número aleatório define a direção que a transações foi gerada. Como uma concessão apenas pode ter duas direções, através da geração de um número aleatório entre 0 a 100, cada direção tem 50% de possibilidade.
- **Tipo de pagamento** - Existem dois tipos de pagamento AVI ou CASH, como já foi referido anteriormente. Ainda no relatório da Illinois Tolway, é evidenciado que 87% das transações são do tipo AVI. Como tal, tendo por base um número aleatório de 0 a 100:

  1) Entre 1 e 87, atribuiu-se a transação o tipo de pagamento *AVI*,
  2) Caso contrário, atribuiu-se a transação o tipo de pagamento *CASH*

- **Altura do dia 2** - existe uma diferenciação de preços aplicados nas configurações base. As opções possíveis são:

  1) *BOTH* - igual para o dia inteiro
  2) *DAYTIME* - entre 6h e as 21h

3) *NIGHTIME* - entre as 22h e as 5h

A primeira opção só ocorre para veículos da classe 1. As restantes classes têm as duas últimas. Como tal, é preciso atribuir primeiro a classe de veículo e, depois verificar a hora em que a transação foi criada.

- **Limitações** - não tem em consideração os seguintes pontos:
  1) Plazas consecutivos - as transações não são geradas em plazas consecutivas, ou seja, a ordem de plazas não é considerada;
  2) Tempo de execução - a criação de transações acarreta muito tempo de execução.

5.3.3 Método de tratamento de dados

Depois de finalizada a recolha, o próximo passo é a análise. A framework de *big data analysis* é a Apache Spark. O objeto JavaRDD é a versão Java da coleção de objetos RDD (*Resilient Distributed Data*). Este tipo de objeto permite efetuar um conjunto de operações em memória. Cada operação corresponde a uma iteração e, o resultado de cada iteração é escrita em memória. Exemplos de operações são mapeamentos, filtragens, uniões, interseções, reduções e agregações de dados. O facto dos dados de entrada serem paralelizados, aumenta significativamente o processamento dos mesmos. Deste modo, as listas de objetos Java necessários serão transformados em objetos JavaRDD.

Operações comuns

- Filtragem de transações ou configurações de preços por:
  1) **Classe de Tarifa** - filtrar com base nas classes de veículo;
  2) **Direção de Saída** - filtrar com base na direção de saída dos veículos;
  3) **Plaza** - filtrar por Plaza de uma determinada concessão;
  4) **Concessão** - filtrar pela autoestrada respetiva.
- Filtragem de eventos meteorológicos, eventos desportivos e feriados;
- Filtragem de *Flow Rate Metrics*;

Na secção seguinte serão evidenciados os processos para o tratamento e aplicação dos dados recolhidos anteriormente, para a criação e treino do modelo de aprendizagem.

5.4 Criação e Treino do modelo de aprendizagem

5.4.1 Preparação do modelo de aprendizagem

A criação do modelo de aprendizagem implica a geração de métricas. Como a classe a aprender é o *flow rate* então, as características associadas a essa mesma classe irão denominar-se *flow rate metrics*. Para criar as métricas é necessário executar os seguintes passos: Para cada iteração de 15 minutos:

1) Analisar a lista de transações recolhida forma a obter o *flow rate* de acordo com o algoritmo de análise do estado de congestionamento de tráfego evidenciado na secção **3.2** .
2) Analisar se existe algum feriado, eventos desportivos e meteorológicos na data corrente e criar os objetos correspondentes: Holiday, List<SportEvent>, MeteoEvents.
3) Consoante o feriado em questão, é relevante atribuir uma nível de importância. São avaliados de 1 a 5.
4) Quanto aos eventos desportivos, apenas interessa saber o número de eventos a decorrer. São considerados eventos de clubes de grande dimensão, não é possível atribuir um nível de importância como no caso dos feriados. Neste caso, apenas se conta o número de eventos a ocorrerem na data corrente.
5) Quanto aos eventos meteorológicos, como é atribuído o valor 0 ou 1, cada evento é considerado individualmente.
6) Por fim, gera-se o objeto *FlowRateMetrics* com toda a informação recolhida, incluindo o mês, dia, hora, minuto e dia de semana da data corrente do *flow rate* calculado, e é armazenado na base de dados. A tabela que representa as *Flow Rate Metrics*, denomina-se FLOW_RATE_METRICS e é representada da seguinte forma.

A Tabela 5.9 contém todas as métricas recolhidas para cada plaza e em intervalo de 15 minutos. Dado que um concessão tem sempre dois sentidos, existe uma *flow rate metrics* para cada *exit point direction*.

Após a geração de todas as *FlowRateMetrics*, estas são agrupadas numa lista. O algoritmo Naive Bayes da biblioteca de machine learning do Spark, utiliza objetos *LabeledPoint* para a aprendizagem. Este objeto é composto por dois elementos: um array de *double's* que armazena as características; e uma *label* que representa a classe. Como as características são representadas num array de *double's* é necessário converter os atributos de cada *FlowRateMetrics* da seguinte forma. A ocorrência de Feriados é representada na forma binária, ou seja, se há um feriado então é atribuído valor "1". A *Label* será necessariamente o atributo *Flow Rate*.

As *LabeledPoint* são agrupadas em objetos JavaRDD. O número de classes a considerar é fundamental para ter a melhor precisão dos resultados. Deste modo, o valor da variável *Flow Rate* será arredondado ao valor da classe mais próxima. Por defeito as classes consideradas são: $400, 800, 1200, 1600, 2000, 2400, 3000, 4000$. Como se pretende avaliar a precisão do algoritmo, na seção de testes, irão ser alteradas as classes ou as características de base. Completo o processo de criação dos dados de treino e de teste, o próximo passo é treinar o modelo.

Campo		Tipo de Dados	Obrigatório	Descrição	Exemplo	Obs.
1	*Id*	Numérico	Sim	Identificador da Configuração da Tarifa	2	PK
2	*Concession Id*	Numérico	Sim	Identificador da Concessão	307	-
3	*Plaza Id*	Varchar2	Sim	Identificador do Plaza	3190	-
4	*Exit Point Direction*	Numérico	Sim	Identificador da direção	2	-
5	*Free Flow Speed*	Numérico	Sim	Velocidade de Fluxo Livre	71.0	-
6	*Month*	Numérico	Sim	Mês	5	-
7	*Day*	Numério	Sim	Dia	1	
8	*Hour*	Numérico	Sim	Hora	12	-
9	*Minute*	Numérico	Sim	Minuto	30	-
10	*Day Of The Week*	Numérico	Sim	Dia da Semana	2	-
11	*Week Number*	Numérico	Sim	Semana do mês	3	-
12	*Holiday*	Varchar2	Sim	Nome do Feriado	"Thanksgiving Day"	-
13	*Sport Events*	Numérico	Sim	Número de eventos desportivos	3	-
14	*Fog*	Numérico	Sim	Nevoeiro	1	-
15	*Rain*	Numérico	Sim	Chuva	0	-
16	*Snow*	Numérico	Sim	Neve	1	-
17	*Hail*	Numérico	Sim	Vento	0	-
18	*Thunder*	Numérico	Sim	Trovoada	0	-
19	*Tornado*	Numérico	Sim	Tornado	0	-
20	*Flow Rate*	Numérico	Sim	Taxa de Fluxo	1090	-

Tabela 5.11: Tabela FLOW_RATE_METRICS e seus atributos.

Campo		Valor	Valor Convertido
1	*Id*	2	2.0
2	*Concession Id*	3062	307.0
3	*Plaza Id*	3190	3190.0
4	*Exit Point Direction*	2	2.0
5	*Free Flow Speed*	71.0	71.0
6	*Month*	5	5.0
7	*Day*	1	1.0
8	*Hour*	12	12.0
9	*Minute*	30	30.0
10	*Day Of The Week*	2	2.0
11	*Week Number*	3	3.0
12	*Holiday*	"Thanksgiving Day"	1.0
13	*Sport Events*	3	3.0
14	*Fog*	1	1.0
15	*Rain*	0	0.0
16	*Snow*	1	1.0
17	*Hail*	0	0.0
18	*Thunder*	0	0.0
19	*Tornado*	0	0.0

Figura 5.2: Exemplo de transformação de uma *FlowRateMetrics* numa *LabeledPoint*

5.4.2 Treino do Modelo de Aprendizagem

A variável *smoothValue* é utilizada como fator de alisamento, ou seja, permite ajustar a função de modo capturar padrões importantes nos dados, deixando de fora valores considerados como ruído.

A imagem 5.3 revela como a biblioteca do Spark permite a criação e treino do algoritmo *Naive Bayes*. O primeiro passo consiste na conversão das *flow rate metrics* em objetos $JavaRDD < LabeledPoint >$, como foi explicado na secção anterior. O segundo passo consiste em dividir os dados em dados de treino e dados de teste. Os dados são divididos de forma aleatória numa proporção de 70%/ 30% para os dados de treino e teste respetivamente. De seguida a função $NaiveBayes.train(...)$ cria o novo modelo *Naive Bayes* tendo por base os dados de treino. Por fim, com auxilio da função $predict(...)$ testa-se se o modelo está a prever com sucesso as classes do dados de teste, devolvendo o par $JavaPairRDD < Double, Double >$ aos pares com os valores previstos e esperados.

```
public static JavaPairRDD<Double, Double> flowRatePrediction(JavaRDD<LabeledPoint> training,
        JavaRDD<LabeledPoint> test, double smoothValue, JavaSparkContext sc, String nameModel) {

    File fileModel = new File(PATH_DIRECTORY_MODELS + nameModel);
    if (fileModel.exists())
        removeModel(fileModel);

    final NaiveBayesModel model = NaiveBayes.train(training.rdd(), smoothValue);
    model.save(sc.sc(), PATH_DIRECTORY_MODELS + nameModel);

    return test.mapToPair(new PairFunction<LabeledPoint, Double, Double>() {
        private static final long serialVersionUID = 1L;

        @Override
        public Tuple2<Double, Double> call(LabeledPoint p) throws Exception {

            return new Tuple2<>(model.predict(p.features()), p.label());

        }
    });
}
```

Figura 5.3: Excerto do código envolvido no treino do modelo de aprendizagem

A figura 5.4 abstrai todos os processos inerentes à fase de criação e treino do modelo de aprendizagem.

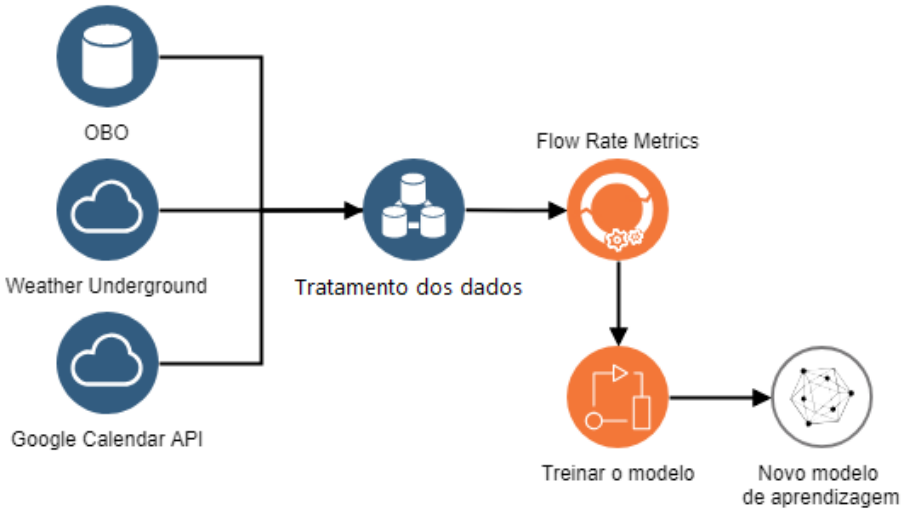

Figura 5.4: Abstração da fase de criação e treino do modelo de aprendizagem

Finalizado o processo de criação e treino, o modelo é armazenado na diretoria principal do projeto para que possa vir a ser utilizado na previsão dos novos preços, como é revelado na secção seguinte.

5.5 Previsão dos novos preços

Finalizado o processo da criação de modelo de aprendizagem, já estão estabelecidas todas as condições para prever os novos preços a serem aplicados em cada plaza para os próximos 15 minutos. De uma forma geral os passos são os seguintes.

1) Recolher a existência de eventos desportivos, meteorológicos e feriados na data corrente.
2) Por plaza
    - Analisar as transações geradas nos últimos 15 minutos; Nesta fase é necessário filtrar as transações por direção de condução e classe de veículo.
    - Filtrar as transações por direção:
        (a) Calcular o *Flow Rate* atual;
        (b) Criar o objeto *Flow Rate Metrics* para os próximos 15 minutos;
        (c) Prever o *flow rate* para os próximos 15 minutos;
        (d) Validar o valor previsto;
        (e) Calcular o nível de serviço associado ao flow rate previsto; 1*
        (f) Atualizar tabelas com configurações de preços: 2*

- Copiar as configurações correntes para a o histórico
- Atualizar as configurações correntes com as previstas para esse momento na iteração anterior
- Atualizar as próximas configurações com os novos valores previstos

(g) Criar um objeto *FlowRateMetrics* com o *flow rate* e guardá-lo para futuros cálculos.

3) Aguardar 15 minutos...

1* Nota: Como já foi referido, o nível de serviço é atribuído com base no *flow rate*. A Tabela 3.4 evidencia a atribuição do nível de serviço mas não inclui o nível F, o qual também é referido. Isto significa que o nível F corresponde a valores de densidade superiores a 45 pc/mi/ln. Além disso, durante a criação do base de conhecimento alguns valores de *flow rate* são muito elevados para a média de valores. Como me baseei em transações da base de dados de Qualidade, é possível que em alguns períodos, para fins de teste, tenham sido geradas mais transações do que seria normal. Como tal, todos os valores de *flow rate* superiores a 4000 pc/h não foram considerados no modelo de aprendizagem.
A atribuição dos preços propriamente dita é realizado da seguinte forma:

- Definiu-se que o nível de de serviço base seria o "C". Deste modo, se o nível de serviço atribuído fosse ou "A"ou "B", é efetuado um desconto de 10% e 5% respetivamente (sobre o valor base definido para a tarifa em questão). Para os níveis de serviço "D", "E"e "F"é efetuada uma penalização de 5%, 10% e 15% respetivamente.

2* Nota: Caso seja a primeira vez que se executa o algoritmo, cria-se as configurações atuais com os valores base, e cria-se a tabela com os próximos preços. As tabelas com as configurações de preços são as seguintes:

- PRICES_CONFIGURATIONS - contém as configurações correntes
- OLD_PRICES_CONFIGURATIONS - contém o histórico de configurações
- NEXT_PRICES_CONFIGURATIONS - contém as próximas configurações

Todas têm a mesma estrutura como tal, apenas irei evidenciar os atributos de uma como exemplo.

A Figura 5.5 abstrai todos os processos inerentes à fase de previsão e atualização dos preços que ocorre a cada 15 minutos.

Campo		Tipo de Dados	Obrigatório	Descrição	Exemplo	Obs.
1	*Id*	Numérico	Sim	Identificador da Configuração da Tarifa	2	PK
3	*Concession Id*	Numérico	Sim	Identificador da Concessão	3062	-
4	*Plaza Id*	Varchar2	Sim	Identificador do Plaza	3190	-
5	*Lane Id*	Varchar2	Não	Identificador da Via	39066	-
6	*Rate Tier*	Varchar2	Sim	Identificador da Classe de Veículo	3	-
7	*Cron Expression Text*	Varchar2	Não	Cron Expression para representar a altura do dia	* * 22-5 * * ? *	-
8	*Rate Type*	Varchar2	Sim	Identificador do Tipo de Pagamento	AVI	-
9	*Price*	Numérico	Sim	Preço da Configuração	105	-
10	*Overwrite Source Price*	Varchar2	Não	-	-	-
11	*Exit Point Direction*	Numérico	Sim	Direção de saída	2	-
12	*Level Of Service*	Varchar2	Sim	Nível de serviço atribuído	"A"	-
13	*Effective Start Date*	Timestamp	Sim	Data de Início da Configuração	15.01.01 06:00:00,0000000 00	-
14	*Effective End Date*	Timestamp	Não	Data de Fim da Configuração	17.01.01 06:00:00,0000000 00	-

Tabela 5.12: Tabela PRICES_CONFIGURATIONS e seus atributos.

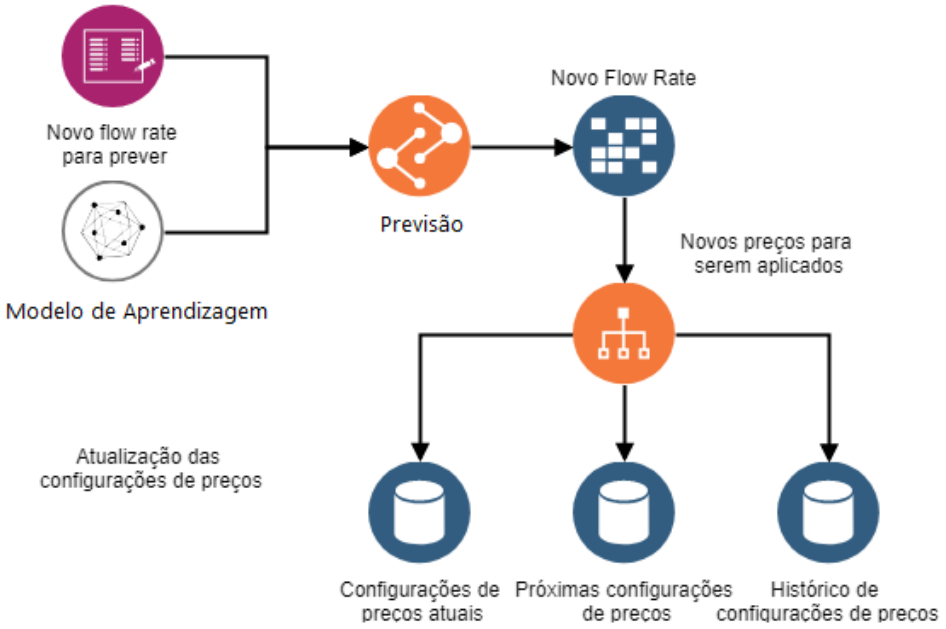

Figura 5.5: Abstração da fase previsão e atualização dos preços

5.6 Visualização e Execução do Sistema

Finalizado o processo de implementação do algoritmo, para que seja possível analisar resultados estatísticos e informativos é necessário visualizá-los. Neste sentido, surgiu a necessidade de criar uma interface gráfica.

Como já foi referido, o sistema está dividido em três camadas, sendo uma delas denominada ***dynamicPricing-web***. É nesta camada que se executa o algoritmo que é implementado na camada denominada ***dynamicPricing-ejb***. A forma como estes pedidos podem ocorrer de várias maneiras. Inicialmente considerei simples *WebServlets* que eram accionados em páginas HTML via formulários. Posteriormente, considerei utilizar a tecnologia *JavaServer Faces*, a qual é utilizada no projeto onde me insiro.

5.6.1 Definição de *JavaServer Faces*

A tecnologia *JavaServer Faces* (JSF) é uma componente do lado do servidor responsável por construir aplicações web em Java.

A tecnologia *JavaServer Faces* fornece um modelo de programação muito bem definido com uma API que representa componentes para lidar com eventos, validações do lado do servidor, conversões de dados, definição de páginas de navegação, etc.. [47]

5.6.2 Implementação da interface gráfica

Inicialmente, como é salientado no requisito não funcional USA-02, pretendia-se que a interface respeitasse os padrões de desenho do site da Illinois Tollway. Contudo, isso implicava importar grande parte do projeto original, o qual tem dependências próprias que não são de todo importantes para a visualização dos dados do algoritmo desenvolvido. Deste modo, foi criada uma interface gráfica própria mais simples.

O padrão de arquitetura de software da camada web que se pretendia utilizar inicialmente foi o *Model-View-Controller* (MVC).

- *Model* - contém a informação representada na *View* e lógica aplicacional que altera a informação no browser consoante a interação do utilizador.
- *View* - exibe a informação ao utilizador, e juntamente com o *Controller* processa a interação do utilizador com o a interface.
- *Controller* - componente responsável pela comunicação entre a *View* e o *Model*.

Em vez disso, foi desenvolvido um modelo mais simplificado em que a lógica aplicacional era executada em controladores, com auxílio de entidades que representam os modelos de dados. Como se utiliza JSF, a interface foi desenvolvida em ficheiros xhtml. Os controladores, através da injeção dos *beans* de sessão da camada de *business*, permitem efetuar todos os pedidos da interação com o utilizador. As entidades utilizadas

são evidenciadas no capitulo 4 no Modelo de Domínio. De seguida são apresentadas as diversas funcionalidades disponíveis.

- **Lista de Concessões** - Permite revelar as características físicas inerentes a cada concessão que são utilizados no algoritmo.
- **Lista de Plazas** - Permite exibir a informação referente a cada plaza de cada concessão.
- **Histórico de Preços Aplicados** - Permite analisar os histórico das configurações de preços com base num conjunto de filtros. Os resultados podem ser apresentados tanto por tabelas ou por gráficos, e são separados em três categorias: preços atuais, próximos preços e histórico de preços. A análise ao histórico de configurações também pode ocorrer a partir de gráficos que têm como *output* preços aplicados ou o nível de serviço associado, no momento em que a configuração foi gerada.
- **Histórico de *Flow Rate Metrics*** - Permite analisar o histórico de métricas que já foram calculados, e que também podem ser filtradas com base num conjunto de filtros. Os resultados também podem ser apresentados tanto por tabelas ou por gráficos.

Desta forma, é possível analisar de alto nível os resultados calculados pela aplicação do algoritmo.

5.7 Sumário

Neste capítulo foram evidenciados todos os processos envolvidos no algoritmo desenvolvido. Todo o trabalho desenvolvido deverá se restringir às capacidades e limitações dessas mesmas ferramentas apresentadas. Como foi possível verificar, o sistema está dividido em duas fases: a fase de aprendizagem e a fase de execução. A fase de aprendizagem envolve a recolha e tratamento dos dados referentes ao ano anterior, de modo a criar o modelo de aprendizagem e a testá-lo posteriormente. A fase de execução envolve a recolha em tempo real da informação necessária para previsão e atualização dos preços atribuídos nos pórticos das autoestradas (concessões). A fase de execução ocorre de forma cíclica a cada 15 minutos de modo a que os preços aplicados possam estar de acordo com o congestionamento de tráfego atual. A visualização de resultados ocorre através do auxilio da tecnologia *Java Server Faces* que permite abstrair os eventos que são invocados em páginas xhtml. No fim de cada ciclo de execução, os dados são atualizados na base de dados para que possam ser reutilizados na iteração seguinte.

Capítulo 6

Testes e Avaliação ao Sistema

O propósito de testar o código desenvolvido é validar se os requisitos e especificações, anteriormente definidos, foram devidamente implementados, e a solução final está de acordo com as expectativas iniciais. É neste sentido que é fundamental efetuar fases de testes de modo a reduzir o risco de se entregar software com erros.

6.1 Definição da estratégia de testes

Existem vários tipos de teste: Testes de Componentes, Testes de Assembly, Testes de Desempenho, Testes de Produto, Testes de Aceitação pelo Utilizador e Testes de Prontidão Operacional.

Tendo em conta, que o projeto consiste num serviço interno ao projeto, sem interação com utilizadores não serão realizados tanto testes de Aceitação pelo utilizador como os testes de Prontidão operacional. Além disso, o tempo estabelecido para realização de testes foi reduzido. Como tal apenas se realizaram testes de desempenho, produto e integração.

Para cada tipo de teste existe um conjunto de cenários de resposta. No caso dos testes de desempenho, os cenários são os seguintes:

1) ***Baseline and Benchmark Test***
   Os *baseline Tests* são os testes que visam capturar dados métricos de desempenho com a finalidade de avaliar a eficácia de mudanças subsequentes de melhoria de desempenho para o sistema ou aplicativo
   Os *Benchmark Tests* é o processo de comparar os resultados dos testes anteriores com um padrão da industria aprovado pela organização.
2) ***Load Test***
   Testes responsáveis por simular o comportamento típico dos utilizadores durante a utilização do sistema condições normais, com foco no número de utilizadores que acedem o servidor, na combinação de transações comerciais executadas no servidor, e o impacto dessas mesmas transações em diferentes componentes do ambiente.

3) ***Volume Test***
   Testes responsáveis por testar o sistema com uma certa quantidade de dados. Em termos gerais, a quantidade de dados pode ser o tamanho da base de dados ou o tamanho do ficheiro de interface que é referente ao teste.
4) ***Stress Test***
   Testes responsáveis por simular os piores cenários de execução com foco em localizar o ponto onde o desempenho do servidor falha.
5) ***Stability Test***
   Testes responsáveis por determinar o desempenho do sistema de forma contínua e a longo prazo.
6) ***Throughput Test***
   Testes responsáveis por assegurar que o sistema pode executar uma determinada operação com base num determinado número de transações num período de tempo especificado.
7) ***Failover Test***
   Testes responsáveis por simular situações em que o sistema falha durante a execução normal.

Os testes de Produto são divididos em duas fases:

1) **Testes de Sistema** (*Application Product Test*)
   Testes responsáveis por verificar se o sistema respeitas as funcionalidades descritas na documentação dos casos de uso, de modo a confirmar que o sistema vai de encontro com os requisitos funcionais, e assegurar que o as configurações do sistema estão corretas. A validação de conteúdos e de formatos não são incluídos nesta fase (Testes unitários).
2) **Testes de Integração**
   Testes responsáveis por verificar se o sistema respeita as funcionalidades dos sistema *end-to-end* descritas nos casos de uso. Tem como objetivo validar os fluxos de execução envolvidos para suportar os requisitos de negócio no sistema.

No contexto do projeto pretende-se essencialmente avaliar a aprendizagem do modelo desenvolvido. Como tal, é relevante realizar uma avaliação qualitativa do modelo. Deste modo, dentro dos cenários de testes de desempenho apenas fará sentido realizar testes de Stress. Para avaliar a execução do sistema fará sentido realizar testes de aplicação e integração.
Em aprendizagem supervisionada existem várias formas para avaliar o desempenho dos algoritmos e dos classificadores. A forma mais comum é a utilização de uma *confusion matrix*, que evidencia a atribuição correta e incorreta de exemplos para cada classe. Cada linha da matriz representa as instâncias de uma classe prevista, enquanto que cada coluna representa as instâncias de uma classe real (ou vice-versa). A matriz pode conter 4 tipos de resultados:

- ***True Positive*** (*tp*) - exemplos corretamente classificados como positivos;
- ***False Positive*** (*fp*) - exemplos negativos incorretamente classificados como positivos;
- ***False Negative*** (*fn*) - exemplos positivos incorretamente classificados como negativos
- ***True Negative*** (*tn*) - exemplos negativos corretamente classificados

No caso de classificação binária, a *confusion matrix* é representada da seguinte forma. As métricas de avaliação de algoritmos de classificação mais amplamente utilizadas são

	Positivo previsto	**Negativo previsto**
Positivo	*tp*	*fn*
Negativo	*fp*	*tn*

Tabela 6.1: *Confusion matrix* para classificação binária

a *precision*, a *recall* e a *Fscore*, as quais são definidas como:

- ***Precison*** - o número de exemplos positivos classificados corretamente dividido pelo número de exemplos classificados pelo sistema como positivos;
- ***Recall*** - o número de exemplos positivos classificados corretamente dividido pelo número de exemplos positivos nos dados.
- ***Fscore*** - relação das anteriores

As medidas de classificação binária utilizam a notação da tabela seguinte.

Medida	**Fórmula**
Accuracy	$\frac{tp+tn}{tp+fn+fp+tn}$
Precision	$\frac{tp}{tp+fp}$
Recall	$\frac{tp}{tp+fp}$
Fscore	$\frac{(\beta^2+1)tp}{(\beta^2+1)tp+\beta^2 fn+fp}$

Tabela 6.2: Medidas de avaliação para classificação binária
[48]

As medidas para a classificação multi-classe baseia-se na classificação binária da seguinte forma: para cada classe C_i: tp_i são os tp; fp_i são os fp; fn_i são os fn; tn_i são os tn.

As medidas de classificação multi-classe utilizam a notação da tabela seguinte. [48] Nota: O valor do parâmetroβ é 1.0 por defeito.

Neste caso, estamos perante um problema de classificação multi-classe. Como tal, é necessário gerar uma *confusion matrix* que relacione todas as classes. O valores das

Medida	Fórmula
Average Accuracy	$\frac{\sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i}}{l}$
Error Rate	$\frac{\sum_{i=1}^{l} \frac{fp_i + fn_i}{tp_i + fn_i + fp_i + tn_i}}{l}$
$\boldsymbol{Precision_M}$	$\frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i}}{l}$
$\boldsymbol{Recall_M}$	$\frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i}}{l}$
$\boldsymbol{Fscore_M}$	$\frac{(\beta^2 + 1) Precision_M Recall_M}{\beta^2 Precision_M + Recall_M}$

Tabela 6.3: Medidas de avaliação para classificação multi-class [48]

classes depende do valor de *Free Flow Speed* à concessão em causa, sendo que as classes correspondem aos valores máximos de *flow rate* para o *Free Flow Speed* da concessão respetiva, como é indicado na Tabela 3.4. Isto significa que, para cada concessão existirá um modelo de aprendizagem associado. Tendo em conta que podem existir valores de *Free Flow Speed* entre os valores evidenciados na tabela 3.4, cada conjunto de classes é representado por intervalos de *Free Flow Speed*, da seguinte forma.

- Para FFS >= 75 mi/h - 820, 1350, 1830, 2170, 2400 e 4000;
- Para 70 <= FFS < 75 mi/h - 720, 1260, 1770, 2170, 2400 e 4000;
- Para 65 <= FFS < 70 mi/h - 710, 1170, 1680, 2090, 2350 e 4000
- Para 60 <= FFS < 65 mi/h - 660, 1080, 1560, 2020, 2300 e 4000;
- Para 55 <= FFS < 60 mi/h - 600, 990, 1430, 1910, 2250 e 4000;

Como já foi referido, o parâmetro *Smooth Value* é utilizado como fator de alisamento. De modo, a verificar qual é o valor do parâmetro que apresenta os melhores resultados, para cada teste, o *smooth value* varia entre 0 a 4000 em intervalos de 100. Para cada valor, o modelo de aprendizagem é treinado 10 vezes. Sendo assim, os resultados associados a cada *smooth value* serão valores médios.

Outro ponto a salientar é que os dados os teste correspondem aos dados de qualidade, como já foi referido e, como os dados não são criados em tempo real, necessitei de atrasar o relógio do computador em pelo menos 3h para obter transações.

Na próxima secção são identificados e documentados todos os testes realizados.

6.2 Identificação de Testes

6.2.1 Testes de Stress

Os testes de stress, no contexto do projeto, centram-se na análise do modelo de aprendizagem em função da variação dos parâmetros de entrada: as características ou a classes. Como tal, cada teste terá ou um conjunto específico de características ou de classes.

Número	Descrição
ST01Test	Características = {concessionId, plazaId, exitPointDirection, ffs, month, day, hour, minute, dayOfWeek, weekNumber, holiday, sportEvents, fog, rain, snow, hail, thunder, tornado}
ST02Test	Características = {ffs, month, day, hour, minute, dayOfWeek, weekNumber, holiday, sportEvents, fog, rain, snow, hail, thunder, tornado}
ST03Test	Características = {month, day, hour, minute, dayOfWeek, weekNumber, holiday, sportEvents, fog, rain, snow, hail, thunder, tornado}
ST04Test	Características = {month, day, hour, minute, dayOfWeek, weekNumber, holiday, sportEvents}
ST05Test	Características = {month, day, hour, minute, dayOfWeek, weekNumber, holiday}
ST06Test	Características = {month, day, hour, minute, dayOfWeek, weekNumber}
ST07Test	Características = {month, hour, minute, dayOfWeek, weekNumber}
ST08Test	Características = {month, hour, minute, dayOfWeek}
ST09Test	Características = {concessionId, plazaId, exitPointDirection, ffs, month, hour, minute, dayOfWeek, weekNumber, holiday, sportEvents, fog, rain, snow, hail, thunder, tornado}
ST010Test	Características = {exitPointDirection, ffs, month, hour, minute, dayOfWeek, weekNumber, holiday, sportEvents, fog, rain, snow, hail, thunder, tornado}
ST11Test	Características = {exitPointDirection, ffs, month, hour, minute, dayOfWeek, holiday, sportEvents, fog, rain, snow, hail, thunder, tornado}
ST12Test	Classes = {400, 800, 1200, 1600, 2000, 2400, 3000, 4000}
ST13Test	Classes = {200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000, 2200, 2400, 3000, 4000}

Tabela 6.4: Tabela com Testes de Stress

6.2.2 Testes de Produto

Testes de Aplicação	
Número	**Descrição**
AT01	Previsão de próximo flow rate
AT02	Atribuição dos novos preços

Tabela 6.5: Tabela com Testes de Aplicação

Testes de Integração	
Número	**Descrição**
IT01	Recolher as transações dos últimos 15 minutos
IT02	Calcular o *Flow Rate* atual
IT03	Gerar Flow Rate Metrics para os próximos 15 minutos
IT04	Prever o *Flow Rate* para os próximos 15 minutos
IT05	Calcular o nível de serviço com base no *Flow Rate* previsto
IT06	Atualizar as tabelas de configurações de preços

Tabela 6.6: Tabela com Testes de Integração

6.3 Documentação dos Testes

6.3.1 Testes de Stress

Todos os testes de stress visam avaliar o modelo de aprendizagem através do calculo das medidas: *Precision*, *Recall*, *Accuracy*, *Error Rate* e Taxa de Erro simples. A última medida apenas verifica o número de casos corretamente previstos. Os tipos de resultados *tp*, *fp*, *fn* e *tn* foram calculados com auxilio de operações da biblioteca da framework Apache Spark. Todos os testes basearam-se numa base de conhecimento composta por 43103 linhas da tabela FLOW_RATE_METRICS.
Todos os resultados obtidos dos testes ST01Test a ST13Test encontram-se em anexo. Para os testes ST01Test a ST011Test os parâmetros de entrada comuns são:

1) ***FreeFlowSpeed*** - 71 mi/h
2) ***Smooth Value*** - 100, 200, ..., 4000
3) **Classes** - 720, 1260, 1770, 2170, 2400 e 4000

Para os testes ST12Test a ST013Test os parâmetros de entrada comuns são:

1) ***FreeFlowSpeed*** - 71 mi/h
2) Todas as ***Flow Rate Metrics*** registadas
3) ***Smooth Value*** - 100, 200, ..., 4000
4) **Características** - concessionId, plazaId, exitPointDirection, ffs, month, day, hour, minute, dayOfWeek, weekNumber, holiday, sportE-vents, fog, rain, snow, hail, thunder, tornado

Os testes *ST01Test* a *ST11Test* são testes semelhantes diferindo nas características utilizadas para o modelo de aprendizagem. Os resultados obtidos são evidenciados por tabelas e o gráfico respetivo. Como já foi referido, para cada teste são analisadas as seguintes métricas: $AverageAccuracy$, $ErrorRate$, $Precision_M$, $Recall_M$, $AverageError$, $StandardDeviation$, $Fscore_M$.

6.3.2 Testes de Aplicação

Parâmetros de entrada comuns:

1) **Plaza Id** - 3003
2) **Concession Id** - 302
3) **Exit Point Direction** - 3

AT01Test

- Descrição: Geração de novas *flow rate metrics* para os próximos 15 minutos e, por sua vez, previsão da classe flow rate associado a essas mesmas métricas. Implica a criação e treino do modelo de aprendizagem anteriormente.
- Ficheiro: *AT01Test.java*
- Parâmetros de Entrada:
    1) **Nome do modelo** - lastYearModel
    2) **Hora do Dia** - 9
    3) **Dia do mês** - 28
    4) **Mês** - 4
    5) **Ano** - 2018
    6) **Características** - concessionId, plazaId, exitPointDirection, ffs, month, day, hour, minute, dayOfWeek, weekNumber, holiday, sportE-vents, fog, rain, snow, hail, thunder, tornado
    7) ***FreeFlowSpeed*** - 71 mi/h
    8) **Classes** - 720, 1260, 1770, 2170, 2400 e 4000
- Resultado Obtido: 720 mi/h
- Resultado Esperado: 720 mi/h

AT02Test

- Descrição: Atualização das configurações de preços. Implica copiar as configurações atuais para o histórico de configurações, atualizar as configurações atuais com as configurações anteriormente previstas e, criar as novas configurações previstas para os próximos 15 minutos. Devolve o número que correspondente à execução do algoritmo. Se for 2, então, o algoritmo foi executado com sucesso.
- Ficheiro: *AT02Test.java*
- Parâmetros de Entrada:
    1) **Level Of Service** - 1
- Resultado Obtido: 2
- Resultado Esperado: 2

6.3.3 Testes de Integração

Parâmetros de entrada comuns:

1) **Plaza Id** - 3003
2) **Concession Id** - 302
3) **Exit Point Direction** - 3

IT01Test

- Descrição: Recolher todas as transações da última hora para um plaza. As transações são recolhidas em períodos de 15 minutos. Verificar se o volume de transações é superior a 0.
- Ficheiro: *IT01Test.java*
- Parâmetros de Entrada:

    1) **Minutos** - 15

- Resultado Obtido: true
- Resultado Esperado: true

IT02Test

- Descrição: Após recolher todas as transações da última hora para um plaza filtradas numa única direção, calcular o *flow rate* atual.
- Ficheiro: *IT02Test.java*
- Parâmetros de Entrada:
- Resultado Obtido: true
- Resultado Esperado: currentFlowRate > 0

IT03Test

- Descrição: Geração das *flow rate metrics* associada aos próximos 15 minutos
- Ficheiro: *IT03Test.java*
- Parâmetros de Entrada:

    1) **Hora do Dia** - 9
    2) **Dia do mês** - 28
    3) **Mês** - 4
    4) **Ano** - 2018

- Resultado Obtido: true
- Resultado Esperado: flm != null

IT04Test

- Descrição: Previsão do *flow rate* associado às *flow rate metrics* dos próximos 15 minutos.
- Ficheiro: *IT04Test.java*
- Parâmetros de Entrada:
  1) **Nome do modelo** - lastYearModel
  2) **Hora do Dia** - 9
  3) **Dia do mês** - 28
  4) **Mês** - 4
  5) **Ano** - 2018
  6) ***FreeFlowSpeed*** - 71 mi/h
  7) **Características** - concessionId, plazaId, exitPointDirection, ffs, month, day, hour, minute, dayOfWeek, weekNumber, holiday, sportE-vents, fog, rain, snow, hail, thunder, tornado
- Resultado Obtido: 720 mi/h
- Resultado Esperado: 720 mi/h

IT05Test

- Descrição: Calculo do nível de serviço para os próximos 15 minutos
- Ficheiro: *IT05Test.java*
- Parâmetros de Entrada:
  1) **Nome do modelo** - lastYearModel
- Resultado Obtido: 3
- Resultado Esperado: 3

IT06Test

- Descrição: Atualização das configurações de preços. Implica copiar as configurações atuais para o histórico de configurações, atualizar as configurações atuais com as configurações anteriormente previstas e, criar as novas configurações previstas para os próximos 15 minutos. Devolve o número que correspondente à execução do algoritmo. Se for 2, então, o algoritmo foi executado com sucesso.
- Ficheiro: *IT06Test.java*
- Parâmetros de Entrada:
  1) **Level Of Service** - 1
- Resultado Obtido: 2
- Resultado Esperado: 2

6.3.4 Observações e Conclusões

ST01Test - ST11Test

Todas as tabelas e respetivos gráficos estão em anexo, no Apêndice B - Resultados dos Testes. A métrica mais representativa dos resultados é o $Fscore_M$, visto que relaciona as métricas $Precision_M$ e $Recall_M$. As métricas $AverageAccuracy$ e $ErrorRate$ são complementares, como tal, apenas fará sentido apresentar os gráficos de uma delas. A métrica $AverageError$ analisa apenas o número de casos classificados corretamente, devolvendo a taxa de erro inerente. Quanto à métrica $StandardDeviation$ é irrelevante dado que os testes apresentam na maioria dos casos valores muito próximos de zero. Isto significa que, os valores previstos são muito próximos para independentemente do valor do *smooth value*. Deste modo, os resultados são apresentados sob a forma de uma tabela e gráficos respetivos relativos ao $Fscore_M$, $AverageAccuracy$ e $AverageError$.

Os resultados dos testes permitiram verificar quais são as *flow rate metrics* que têm mais peso no treino do modelo de aprendizagem. Sendo o *Naive Bayes* um algoritmo probabilístico, as métricas que têm mais peso são as que têm mais probabilidade de acontecer. Um fator importante na atribuição de probabilidades é o número de vezes que o valor se repete, logo, o fator com valores constantes irá atribuir valores de probabilidade mais elevados.

No caso do problema em questão, apenas se teve em consideração os dados referentes a um único plaza de um concessão. Além disso, os dados de qualidade apenas correspondiam a uma única direção de circulação, e o campo *free flow speed* também era constante. Isto significa que os campos *concessionId*, *plazaId*, *exitPointDirection* e *ffs* são sempre iguais. Como tal, é expectável que o teste ST01Test apresente os melhores resultados, exceptuando em relação ao teste ST09Test. Neste caso, não se teve em conta o campo *day*. A título de exemplo, se este campo tiver como o valor "1"existem 1534 resultados que corresponde a apenas 3% da base de conhecimento, logo não contribui de forma positiva para a aprendizagem do modelo. O melhor resultado de $Fscore_M$ referente ao teste ST01Test foi 0.380 mas trata-se de um *outlier*, logo não pode ser considerado. Deste modo, para o teste ST01Test o melhor resultado foi 0.355 enquanto que no caso do teste ST09Test foi 0.358.

A partir do teste ST03Test não são considerados os campos com valores constantes, sendo possível verificar uma redução significativa das métricas $Fscore_M$, $AverageAccuracy$ e $AverageError$. Mesmo com a variação do *Smooth Value* os resultados são sempre inferiores em qualquer métrica em análise.

Os campos *holiday*, *sportEvents* e os eventos meteorológicos revelaram ser pouco significativos dado que os resultados são muito idênticos quando deixam de ser considerados, como é possível verificar nos testes ST04Test, ST05Test e ST06Test, respetivamente.

O teste ST07Test apresentou resultados pouco esclarecedores, dado que, à medida que

o *smooth value* aumenta, o $Fscore_M$ varia de forma irregular.

O teste ST08Test apresentou um comportamento muito diferente dos restantes. À medida que se aumenta o valor de *smooth value*, a $AverageAccuracy$ diminui e, por sua vez, a $AverageError$ aumenta. No caso do $Fscore_M$, verifica-se um decréscimo mais significativo em relação aos outros testes. Isto significa que o campo *weekNumber* tem mais peso do que as restantes.

Os teste ST10Test e ST11Test apresentam resultados semelhantes aos do teste ST02Test, embora ligeiramente inferiores.

De uma forma geral, a $AverageAccuracy$ obtém melhores resultados com o aumento do *smooth value*, e por sua vez, $AverageError$ diminui. Isto significa que, em média, o número de casos corretamente classificados aumenta com a variação do *smooth value*, embora em alguns casos não seja evidente, dado que a variação é muito reduzida.

Concluindo, tendo em conta a base de conhecimento, o melhor caso será ter em consideração todos os campos exceptuado o campo *day*.

Os valores obtidos não são de todo satisfatórios, mas tendo em conta que se trata de uma prova de conceito seria de esperar que os resultados ficassem à quem das expectativas. A aplicação de ML no sector das portagens trata-se de um conceito novo e, que com o devido estudo aprofundado, tornar-se-á uma tecnologia muito viável no futuro.

ST12Test - ST13Test

Em relação aos testes em que se variavam o número de classes, ficou claro que quanto maior é o número de classes pior são os resultados. Para os testes anteriormente evidenciados são consideradas apenas 6 classes. Em qualquer teste apresentado o valor médio de $Fscore_M$ foi sempre superior a 0.24. O teste ST12Test considera 8 classes e apresentou resultados entre 0.05 e 0.227. O teste ST13Test considera 14 classes e apresentou resultados entre 0.02 e 0.160. Como tal, é possível concluir que o aumento do número de classes tem um efeito negativo nos resultados.

6.3.5 Testes de Produto

Tanto os testes de Aplicação como de Integração foram executados com sucesso. Isto significa que o sistema apresenta o comportamento esperado.

6.4 Cobertura dos requisitos

Tendo em conta os testes realizados é possível verificar quais foram os requisitos identificados inicialmente que foram cobertos pelos testes.

- **Requisitos de Negócio**

  1) **Cobertos**: Todos

2) **Não Cobertos**: -

- **Requisitos Operacionais**
  1) **Cobertos**: RO-02
  2) **Não Cobertos**: RO-01
- **Requisitos não funcionais**
  1) **Cobertos**: ESC-02, ESC-03, DES-01, INT-01, DIS-01 USA-02, REC-01 e RES-01
  2) **Não Cobertos**: ESC-01, DES-02, USA-01, MAN-01 e SEG-01

6.4.1 Justificação para os requisitos não cobertos

- **RO-01**
  A taxa de erro mínima verificada foi 38%. Isto significa nem sempre o valor do flow rate previsto é aceitável.
- **ESC-01**
  Não se executou o algoritmo de forma contínua durante 1 dia e para mais do que um plaza. Como tal, não é possível concluir que o sistema consegue lidar com pelo menos 3.5 milhoes de transações diárias
- **DES-02**
  Tal como foi justificado no caso do RO-01, a taxa de erro mínima verificada foi 38%.
- **USA-01**
  Tendo em conta à elevada complexidade do projeto, seria muito difícil adaptar o sistema desenvolvido com a interface do site da Illinois Tollway. A interface tem por defeito muitas dependências às quais não é possível responder. Caso o sistema desenvolvido venha a ser integrado no projeto global, nesse caso faz mais sentido adaptar a interface e não o contrário. Deste modo, foi desenvolvido uma interface simplificada mas que evidencia os dados relevantes.
- **MAN-01**
  Não foram programados quaisquer mecanismos para garantir a manutenção do sistema.
- **SEG-01**
  Não existe qualquer tipo de mecanismo de segurança no algoritmo porque para fins de projeto de tese não faria sentido o seu desenvolvimento.

6.5 Cobertura dos casos de uso e KPI's

Todos os casos de uso que foram identificados anteriormente foram implementados no sistema, significa que todas as tarefas propostas inicialmente foram realizadas. Para ana-

lisar a cobertura dos KPI's seria necessário colocar o sistema em funcionamento numa situação real e verificar o impacto que teria no normal funcionamento da infraestrutura rodoviária.

6.6 Análise dos resultados

De uma forma geral posso concluir que os objetivos iniciais foram realizados com sucesso. O sistema desenvolvido é capaz de prever a atribuição de preços no futuro com base num modelo de aprendizagem e em dados recolhidos em tempo real.

No que toca ao desempenho do modelo de aprendizagem os resultados não são de todo satisfatórios, visto que não se obteve resultados de $Fscore_M$ superiores a 0.380 (numa escala de 0 a 1). Ficou claro que o número de classes consideradas tem um efeito direto nos resultados, dado que, quanto mais classes se consideram piores são os resultados. No que toca às características, a variação dos resultados não foi muito acentuada. Ficou claro que as características que podem ter uma largo intervalo de valores não contribuem para o desempenho do algoritmo. Sendo o *Naive Bayes* um algoritmo probabilístico, a probabilidade atribuída a cada característica está diretamente relacionada com a gama de valores possíveis. Como tal, será interessante adequar as características de modo a que tenham um intervalo de valores reduzidos. Portanto, ou o algoritmo escolhido não é o mais eficaz ou a forma como o modelo foi criado não é de todo o mais eficiente, implicando assim identificação de novas classes e características e ajustamento na gama de valores.

Deste modo, é possível concluir que embora o sistema desenvolvido respeite os pressupostos iniciais, o método escolhido para aprendizagem não foi o mais indicado.

Capítulo 7

Conclusão

Neste capítulo irei resumir brevemente todo o trabalho desenvolvido, incluindo naturalmente as principais contribuições e competências adquiridas, as dificuldades encontradas e os pontos a melhorar no futuro.

7.1 Principais Contribuições

O projeto de tese teve como objetivo desenvolver um serviço de *dynamic pricing* com aprendizagem automática no setor das portagens que possibilitasse a diminuição do congestionamento de tráfego, maximizando o lucro obtido.

Após o levantamento do trabalho relacionado, de modo a evidenciar as várias formas de *Congestion Pricing*, *Machine Learning* e Análise de Big Data, foi possível realizar o algoritmo funcional como prova de conceito para o problema em questão.

Tendo por base a tese de João Gomes que já tinha abordado o mesmo tema, desenvolvi um algoritmo que é capaz de prever o estado de tráfego num determinado ponto, com base em informações recolhidas em tempo real e no modelo de aprendizagem construído.

Para a desenvolvimento do serviço foram integradas várias *frameworks*: a Apache Spark para as componentes de *Big Data* e *Machine Learning*; o Java Server Faces para a componente web, o JBoss como servidor aplicacional, e o JUnit com Arquillian como *framework* de teste. Todos os projetos java desenvolvidos são projetos Maven para facilitar a importação de dependências.

O algoritmo implementado é divido em 2 fases distintas: Criação e Treino do modelo de aprendizagem, e Previsão e Atribuição de preços. Ambas recorreram à utilização da *framework* Apache Spark, que contém uma biblioteca bastante extensa para tratamento de Big Data. Como já foi referido, o cálculo do nível de serviço baseou-se no algoritmo descrito no livro ”Traffic and Highway Engineering”de Nicholas J.Garber e Lester A. Hoel. A fase de previsão baseou-se no algoritmo *Naive Bayes* da biblioteca de ML da Apache Spark. O modelo de aprendizagem é baseado num sistema de classificação e previsão praticamente em tempo real. A métrica prevista é sempre verificada para garantir

um grau de confiança elevado na atribuição de preços.

O serviço foi testado com dados reais fornecidos pelo projeto onde fui inserido e por API's externas (*Weather Underground e Google Calendar*), de modo a obter resultados mais fidedignos.

O serviço desenvolvido representa uma prova de conceito funcional, dado que apenas foi testado para um único plaza de uma concessão, mas está preparado para analisar todos os plazas de uma determinada concessão. Devido à falta de tempo não foi possível realizar os testes respetivos.

7.2 Competências Adquiridas

O projeto de tese permitiu-me ter contacto com novas tecnologias que se tornaram fundamentais para a implementação do serviço de modo a cumprir com os requisitos definidos inicialmente. Para o tratamento de dados analisei em detalhe as *frameworks* Storm e Spark da Apache, tendo posteriormente escolhido o Spark por se adequar melhor ao problema em mãos. Tive a oportunidade de aprofundar os meus conhecimentos de ML e a sua aplicação num caso prático, utilizando a biblioteca de ML do Spark.

Como desenvolvi um projeto *enterprise application*, tive oportunidade de rever alguns conhecimentos fundamentais de Java EE que foram abordados em cadeiras de anos anteriores, incluindo necessariamente o processo de injeção de dependências, configuração de um servidor aplicacional e acesso à base de dados (neste caso via Hibernate).

Além disso, pelo facto de ter realizado o projeto de tese numa empresa, tive o privilégio de observar o funcionamento de um projeto que segue uma metodologia Agile.

7.3 Dificuldades encontradas

A implementação do algoritmo acarretou algumas dificuldades dado que algumas tecnologias utilizadas, como é o caso do Spark, terem praticamente toda a sua documentação atual em Java 8, e o projeto onde fui inserido apenas utilizar o Java 7. Isto implicou um esforço adicional em procurar soluções compatíveis com as configurações do projeto. O facto de também nunca ter implementado um projeto *enterprise application* de raiz, acarretou mais trabalho, mas ao mesmo tempo tornou o processo mais aliciante.

O treino e teste de modelo de aprendizagem implicou um esforço adicional de modo a obter melhores resultados. Como os meus conhecimentos de ML eram reduzidos necessitei de fazer uma pesquisa mais aprofundada para entender melhor o conceito.

Uma condição fundamental para a execução do algoritmo é aceder a transações das portagens em tempo real. Para isso, precisei de ter acesso a dados de produção ou de qualidade do próprio projeto. Como os acessos apenas me foram concedidos já em fase adiantada da implementação do serviço, foi necessário gerar dados de teste através de um

simulador de transações com um determinado grau de aleatoriedade para obter resultados mais realistas. Isto significa, que sempre que precisei de executar o algoritmo teve de criar novas transações na ordem das dezenas de milhar. Este processo foi bastante lento, como tal, acarretou mais tempo para a implementação e teste do serviço.

Além disso, em meados de Abril, após uma reunião de equipa, pude constatar que não estava a cumprir os requisitos iniciais. Como tal, foi necessário reescrever boa parte da implementação e relatório do mesmo. Consequentemente, o planeamento do projeto foi alterado.

7.4 Trabalho Futuro

Quando já me encontrava numa fase final da implementação do serviço, deram-me a conhecer a intenção de migrar o projeto corrente para Java 8. Para manter a coerência com o projeto seria interessante adaptar o código para a versão seguinte. Além disso, o Java 8 introduziu a programação funcional que melhora substancialmente a legibilidade do código, Hoje em dia, o Apache Spark tem toda a sua documentação em Java 8. Como tal, a migração traria benefícios não só em termos de legibilidade, como também em termos de apoio na documentação.

Além disso, pontos a realizar no futuro são:

- Ter em consideração as férias escolares no algoritmo de previsão;
- Ter em consideração transações que foram armazenadas com atraso;
- Ter em consideração acidentes que ocorram em tempo real;
- Ter em consideração a localização geográfica de cada plaza;
- Analisar o congestionamento a partir de múltiplos sensores localizados entre plazas;
- Estender o conceito a toda a infraestrutura rodoviária da Illinois Tollway;
- Adaptar o conceito ao sistema de portagens da Ascendi.

Outro ponto importante passa por redefinir a gama de valores utilizados nas características já utilizadas de modo a alcançar melhores resultados. Deste modo, seria possível utilizar o modelo *Naive Bayes* de forma mais eficiente.

Bibliografia

[1] C. R. Lindsey and E. T. Verhoef, "Traffic Congestion and Congestion Pricing," Tinbergen Institute, Amsterdam and Rotterdam, Tech. Rep., 2000.

[2] M. Minelli, M. CHambers, and A. Dhiraj, "Big Data Analytics," 2013.

[3] A. B. Ferreira, "Lisboa vai ter 84 mil lugares de estacionamento pago," 2017. [Online]. Available: https://www.dn.pt/sociedade/interior/lisboa-vai-ter-84-mil-lugares-de-estacionamento-pago-8656670.html

[4] P. T. Decorla-souza, M. F. Muriello, J. N. Buxbaum, D. Blackledge, D. Toups, A. Greenberg, and T. Higgins, *TR NEWS Pricing Road Use to Address Congestion 23 What ' s in Store for Second-Generation Express Lanes in San Diego*, 2009, no. 263.

[5] J. Gomes, "Sistema de pricing dinâmico de cobrança de portagens," Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2016.

[6] Brisa, "Meios de pagamento." [Online]. Available: http://www.brisaconcessao.pt/content.aspx?menuid=35{&}eid=71

[7] A. de Palma and R. Lindsey, "Technologies, Traffic congestion pricing methodologies and," 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X11000362

[8] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (ALPR): A state-of-the-art review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 2, pp. 311–325, 2013.

[9] R. S. Matos, "Solução integrada de video-based vehicle identification," Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2013.

[10] E. F. Morgul, "Simulation Based Evaluation of Dynamic Congestion Pricing Algorithms and Strategies," 2010.

[11] I. Tollway, "Superior Customer Service." [Online]. Available: https://www.illinoistollway.com/tolling-information{#}SuperiorCustomerService

[12] D. J. Walker, P. P. Blythe, and A. Pickford, "Congestion charging: learning the lessons and moving forward Dr," *Fish Diseases and Disorders*, 2013.

[13] "Impacts monitoring 2007," *Transport for London*, vol. 3, no. July, p. Imagen PDF, 2007.

[14] R. Mason, "London to introduce £10 vehicle pollution charge, says Sadiq Khan," 2017. [Online]. Available: https://www.theguardian.com/environment/2017/feb/17/london-to-introduce-vehicle-pollution-charge-in-october-says-mayor-sadiq-khan

[15] W. Elmaghraby and P. Keskinocak, "Dynamic Pricing in the Presence of Inventory Considerations : Directions," *Management Science*, vol. 49, no. 10, pp. 1287–1309, 2003. [Online]. Available: http://mansci.journal.informs.org/cgi/doi/10.1287/mnsc.49.10.1287.17315

[16] Uber, "O que é uma tarifa dinâmica?" [Online]. Available: https://help.uber.com/h/34212e8b-d69a-4d8a-a923-095d3075b487

[17] J. Supernak, C. Kaschade, and D. Steffey, "Dynamic Value Pricing on I-15 in San Diego: Impact on Travel Time and Its Reliability," 2003. [Online]. Available: http://trrjournalonline.trb.org/doi/abs/10.3141/1839-04

[18] S. Shepard, "Traffic management sensor innovations: the Dutch experience," 2018.

[19] T. O. Ayodele, "Machine Learning Overview." [Online]. Available: http://www.intechopen.com/books/salmonella-a-dangerous-foodborne-pathogen/attachment-and-biofilm-formation-by-salmonella-in-food-processing-environments{%}0AInTech

[20] R. Giryes and M. Elad, "Reinforcement Learning: A Survey," *Eur. Signal Process. Conf.*, pp. 1475–1479, 2011.

[21] H. E, M. J, and S. P, "Experiments in Induction," taylorfrancis.com, Tech. Rep., 1966.

[22] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica*, vol. 31, pp. 249–268, 2007.

[23] B. L., F. J.H., O. R.A., and S. C.J., "Classification and Regression Trees," *Wadsforth International Group*, 1984.

[24] J. R. Quinlan, "Discovering rules by induction from large collections of examples," *Expert Systems in the Microelectronic age*, pp. 168–201, 1979.

[25] J. Furnkranz, “Separate-and-Conquer Rule Learning. Artificial Intelligence Review,” pp. 3–54, 1999.

[26] M. Negnevitsky, *Artificial Intelligence*, 2005. [Online]. Available: www.pearsoned.co.uk

[27] N. J. Nilsson, “Learning machines,” PsycINFO, Tech. Rep., 1965.

[28] T. Cover and P. Hart, “Nearest neighbor pattern classification,” no. IEEE Transactions on Information Theory, pp. 13(1): 21–7, 1967.

[29] T. Abeel, Y. V. D. Peer, and Y. Saeys, “Java-ML: A Machine Learning Library,” *Journal of Machine Learning Research*, vol. 10, pp. 931–934, 2009.

[30] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham, “Weka: Pratical Machine Learning Tools and Techniques with Java Implementations,” waikato.researchgateway.ac.nz, Tech. Rep., 1999.

[31] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, “MULAN: A Java library for multi-label learning,” Cheng Soon Ong, Tech. Rep., 2011.

[32] P. Russom, “TDWI Best Practices Report: Big Data Analytics,” The Data Warehouse Institute (TDWI), Tech. Rep., 2011.

[33] A. Ribeiro, André e Rodrigues da Silva, “Data Modeling and Data Analytics: A Survey from a Big Data Perspective,” *Journal of Software Engineering and Applications*, vol. 08, no. 12, pp. 617–634, 2015. [Online]. Available: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/jsea.2015.812058

[34] Hadoop, “What is Hadoop.” [Online]. Available: https://hadoop.apache.org/

[35] T. da Silva Morais, “Survey on Frameworks for Distributed Computing,” pp. 95–105, 2015.

[36] A. S, “Frameworks for Distributed Computing,” Tech. Rep., 2014.

[37] M. Zaharia, M. Chowdhury, T. Das, and A. Dave, “Fast and Interactive Analytics over Hadoop Data with Spark,” *Usenix*, vol. 37, no. 4, pp. 45–51, 2012.

[38] Apache, “Apache Spark.” [Online]. Available: https://spark.apache.org/

[39] J. Bez, “Plataformas de Big Data: Spark, Storm e Flink,” no. July, 2015.

[40] Apache, “Apache Incubator - S4.” [Online]. Available: http://incubator.apache.org/projects/s4.html

[41] "Storm@twitter," *Proceedings of the 2014 ACM SIGMOD international conference on Management of data - SIGMOD '14*, pp. 147–156, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2588555.2595641

[42] Apache, "Apache Storm." [Online]. Available: http://storm.apache.org/

[43] N. J. Garber and L. A. Hoel, *Traffic and Highway Engineering*, 2009.

[44] Oracle and O. its affiliates, "The Java EE 6 Tutorial." [Online]. Available: https://docs.oracle.com/javaee/6/tutorial/doc/bnaph.html

[45] H. Zhang, "The Optimality of Naive Bayes," *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference FLAIRS 2004*, vol. 1, no. 2, pp. 1–6, 2004. [Online]. Available: http://www.aaai.org/Papers/FLAIRS/2004/Flairs04-097.pdf

[46] JBoss, "Arquillian: An integration testing framework for Java EE." [Online]. Available: https://docs.jboss.org/arquillian/reference/1.0.0.Alpha1/en-US/html

[47] M. Colsch, "Illinois Tollway Comprehensive Study Update Dear," Tech. Rep., 2017.

[48] A. Left and T. Rayfield, "Web-Application Development Using the ModelNiewl-Controller Design Pattern," ieeexplore.ieee.org, Tech. Rep., 2001.

[49] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.ipm.2009.03.002

[50] P. Chandarana and M. Vijayalakshmi, "Big Data Analytics Frameworks," Proceedings of the International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA), Mumbai, Tech. Rep., 2014. [Online]. Available: http://dx.doi.org/10.1109/cscita.2014.6839299{%}0A[36]Poole,J.,Chang,D.,Tolbert,D.andMellor,(2002)Common

[51] G. D. F. Morales and A. Bifet, "SAMOA: Scalable Advanced Massive Online Analysis," *Journal of Machine Learning Research*, vol. 16, pp. 149–153, 2015. [Online]. Available: http://jmlr.org/papers/v16/morales15a.html

[52] T. Graves and B. Evans, "Spark and Storm at Yahoo."

[53] "Dynamic Pricing with Online Learning and Strategic Consumers: An Application of the Aggregating Algorithm," *Operations Research*, vol. 57, no. 2, pp. 327–341, 2009.

[54] N. Kourtellis, G. De Francisci Morales, A. Bifet, and A. Murdopo, "VHT: Vertical hoeffding tree," *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, pp. 915–922, 2016.

[55] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2013, vol. 53, no. 9.

[56] E. Miranda, "Release Planning & Buffered MoSCoW Rules," no. September, pp. 1–9, 2013.

Apêndice A

Fatores de Ajustamento do *Free Flow Speed*

	Tipo de Terreno		
Fator	**Plano**	**Ondulado**	**Montanhoso**
E_T	1.5	2.5	4.5
E_R	1.2	2.0	4.0

Tabela A.1: Nº de Veículos de passageiros equivalentes para veículos pesados (E_T) e RVs (E_R) em segmentos de autoestrada. Adaptado de (Garber & Hoel, 2009)

	Redução de *Free-Flow Speed*, f_{LC} (mi/h)			
	Nº de vias numa só direção			
Margem direita da via (ft)	2	3	4	5
≥6	0.0	0.0	0.0	0.0
5	0.6	0.4	0.2	0.1
4	1.2	0.8	0.4	0.2
3	1.8	1.2	0.6	0.3
2	2.4	1.6	0.8	0.4
1	3.0	2.0	1.0	0.5
0	3.6	2.4	1.2	0.6

Tabela A.2: Fator de ajustamento para a largura da margem direita da via. Adaptado de (Garber & Hoel, 2009)

Nº de vias numa só direção	Redução de *Free-Flow Speed*, f_N (mi/h)
≥ 5	0.0
4	1.5
3	3.0
2	4.5

Tabela A.3: Fator de ajustamento para o nº de vias. Adaptado de (Garber & Hoel, 2009)

Interchanges por milha	Redução de *Free-Flow Speed*, f_{ID} (mi/h)
≤ 0.50	0.0
0.75	1.3
1.00	2.5
1.25	3.7
1.50	5.0
1.75	6.3
2.00	7.53

Tabela A.4: Fator de ajustamento para a densidade na zona de *interchange*. Adaptado de (Garber & Hoel, 2009)

Largura da via (ft)	Redução de *Free-Flow Speed*, f_{LW} (mi/h)
12	0.0
11	1.9
10	6.6

Tabela A.5: Fator de ajustamento para a largura da via. Adaptado de (Garber & Hoel, 2009)

Apêndice B

Resultados dos Testes

ST01Test

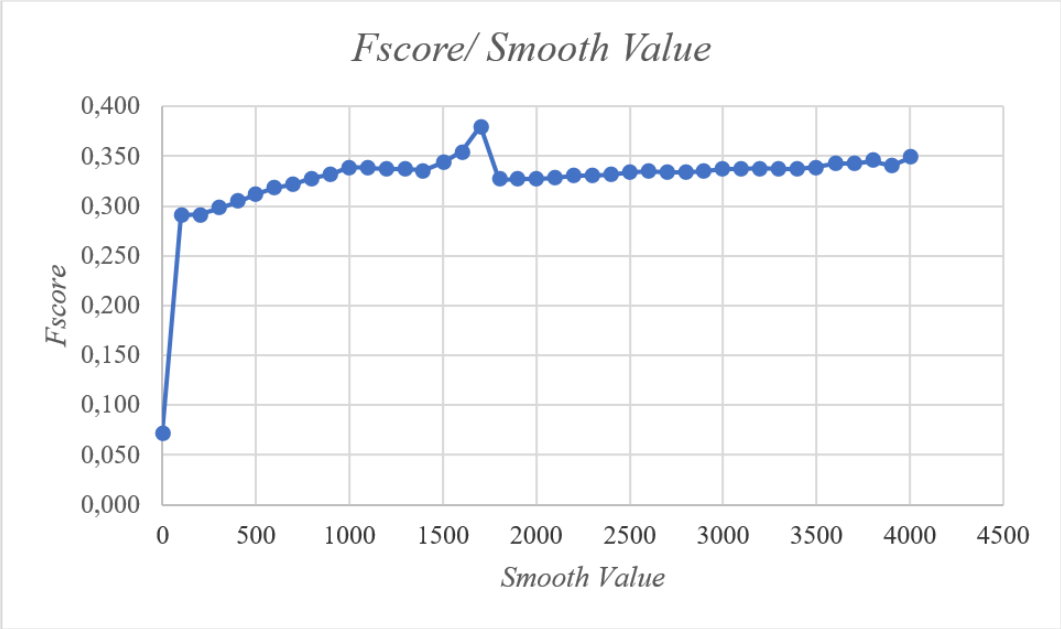

Figura B.1: ST01Test - Gráfico *Fscore/ Smooth Value*

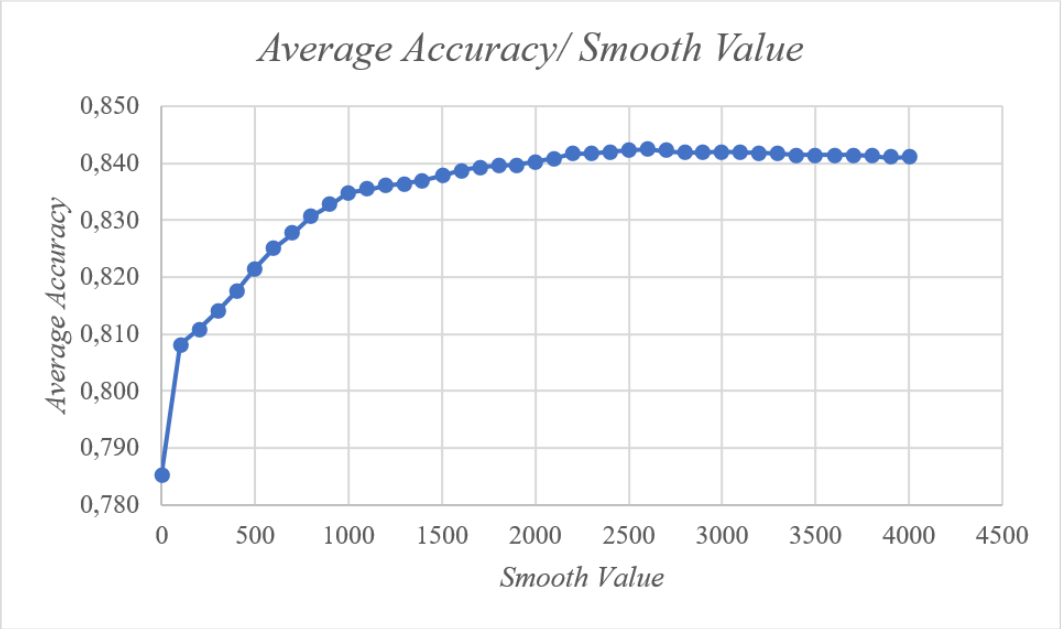

Figura B.2: ST01Test - Gráfico *Average Accuracy/ Smooth Value*

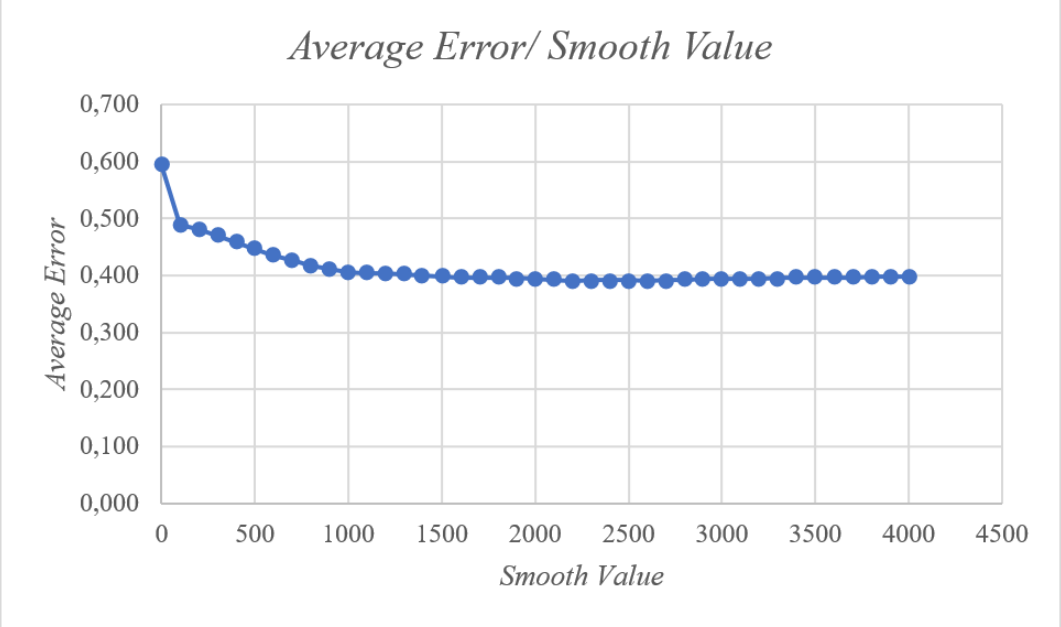

Figura B.3: ST01Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,051	0,125	0,215	0,785	0,595	0	0,072
100	0,289	0,293	0,192	0,808	0,490	0	0,291
200	0,289	0,294	0,189	0,811	0,481	5,55112E-17	0,292
300	0,296	0,300	0,186	0,814	0,470	0	0,298
400	0,303	0,306	0,182	0,818	0,459	0	0,304
500	0,310	0,314	0,178	0,822	0,447	5,55112E-17	0,312
600	0,315	0,321	0,175	0,825	0,436	0	0,318
700	0,319	0,324	0,172	0,828	0,428	1,11022E-16	0,321
800	0,327	0,329	0,169	0,831	0,418	0	0,328
900	0,332	0,331	0,167	0,833	0,412	0	0,331
1000	0,345	0,334	0,165	0,835	0,406	1,11022E-16	0,339
1100	0,344	0,333	0,165	0,835	0,405	0	0,339
1200	0,341	0,333	0,164	0,836	0,403	5,55112E-17	0,337
1300	0,343	0,332	0,164	0,836	0,402	0	0,338
1400	0,341	0,331	0,163	0,837	0,401	5,55112E-17	0,336
1500	0,358	0,331	0,162	0,838	0,399	0	0,344
1600	0,382	0,331	0,161	0,839	0,397	5,55112E-17	0,355
1700	0,447	0,330	0,161	0,839	0,396	5,55112E-17	0,380
1800	0,324	0,329	0,160	0,840	0,396	5,55112E-17	0,327
1900	0,325	0,328	0,160	0,840	0,395	0	0,327
2000	0,327	0,328	0,160	0,840	0,394	0	0,327
2100	0,330	0,327	0,159	0,841	0,393	5,55112E-17	0,328
2200	0,334	0,327	0,158	0,842	0,391	5,55112E-17	0,330
2300	0,336	0,326	0,158	0,842	0,392	5,55112E-17	0,331
2400	0,340	0,325	0,158	0,842	0,392	5,55112E-17	0,332
2500	0,344	0,324	0,158	0,842	0,392	5,55112E-17	0,334
2600	0,348	0,323	0,157	0,843	0,391	5,55112E-17	0,335
2700	0,348	0,322	0,158	0,842	0,392	5,55112E-17	0,334
2800	0,349	0,320	0,158	0,842	0,393	5,55112E-17	0,334
2900	0,353	0,319	0,158	0,842	0,393	5,55112E-17	0,335
3000	0,359	0,318	0,158	0,842	0,394	5,55112E-17	0,337
3100	0,361	0,316	0,158	0,842	0,394	0	0,337
3200	0,364	0,315	0,158	0,842	0,395	0	0,338
3300	0,364	0,314	0,158	0,842	0,395	0	0,337
3400	0,366	0,313	0,158	0,842	0,396	5,55112E-17	0,337
3500	0,372	0,312	0,158	0,842	0,396	5,55112E-17	0,339
3600	0,381	0,311	0,158	0,842	0,396	5,55112E-17	0,342
3700	0,384	0,310	0,159	0,841	0,397	5,55112E-17	0,343
3800	0,392	0,309	0,159	0,841	0,397	5,55112E-17	0,345
3900	0,383	0,307	0,159	0,841	0,398	0	0,341
4000	0,405	0,306	0,159	0,841	0,398	0	0,349

Tabela B.1: Resultados do teste ST01Test

ST02Test

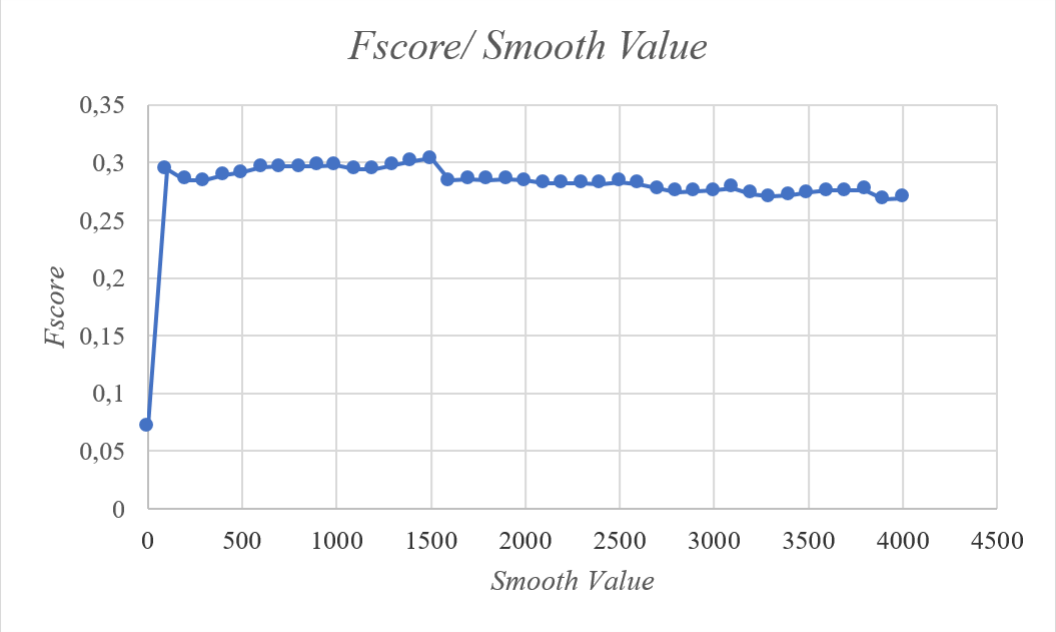

Figura B.4: ST02Test - Gráfico *Fscore/ Smooth Value*

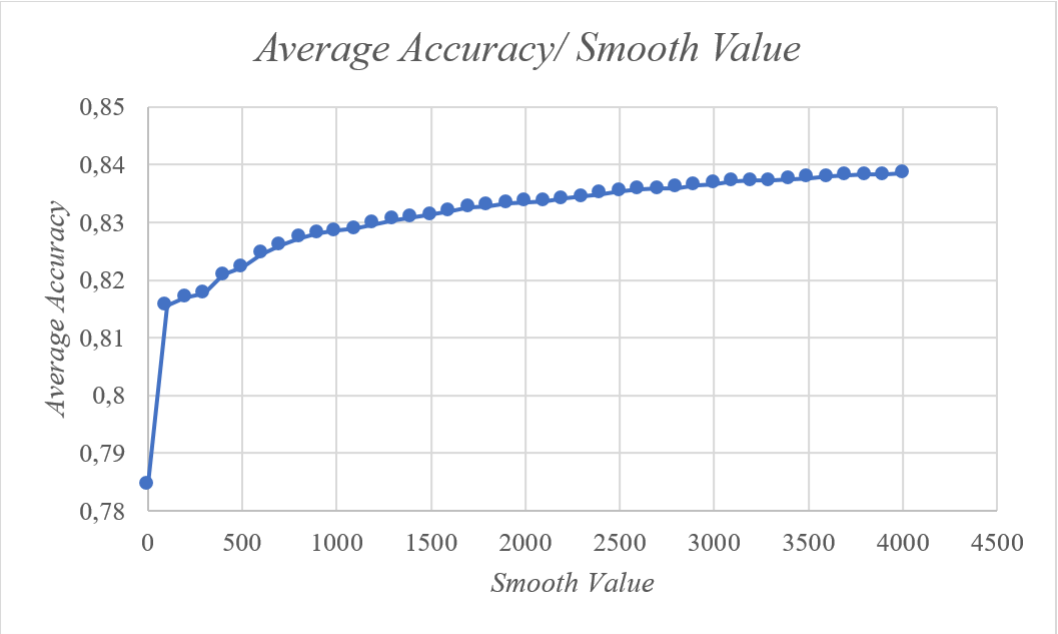

Figura B.5: ST02Test - Gráfico *Average Accuracy/ Smooth Value*

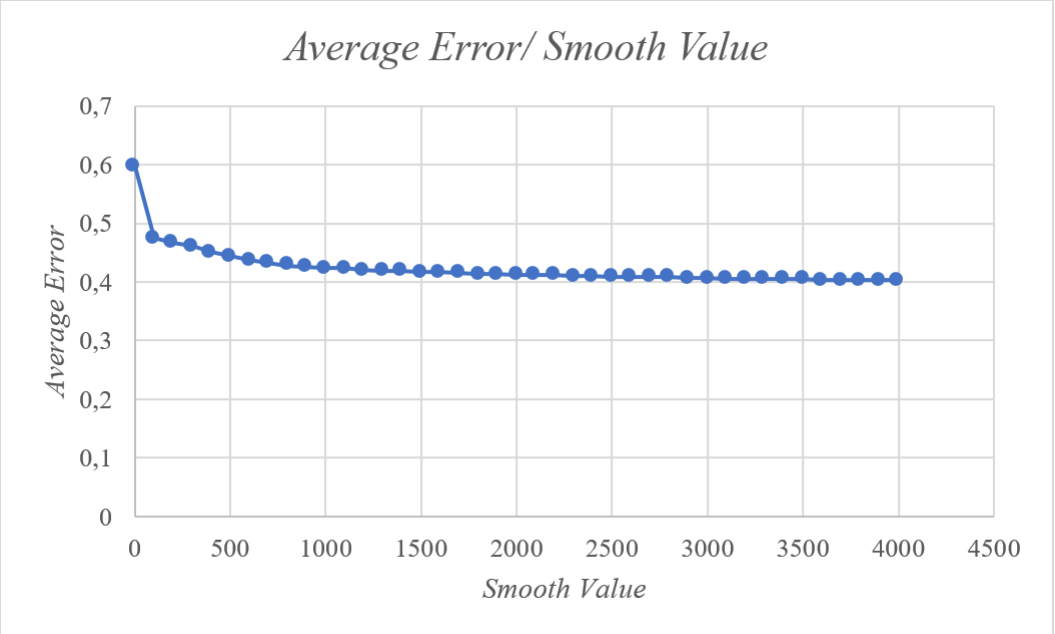

Figura B.6: ST02Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,050	0,125	0,215	0,785	0,598	0	0,072
100	0,285	0,304	0,184	0,816	0,475	0	0,294
200	0,275	0,295	0,183	0,817	0,466	0	0,285
300	0,274	0,296	0,182	0,818	0,462	5,55112E-17	0,285
400	0,278	0,302	0,179	0,821	0,451	1,11022E-16	0,289
500	0,279	0,304	0,178	0,822	0,445	0	0,291
600	0,284	0,308	0,176	0,824	0,437	0	0,296
700	0,284	0,310	0,174	0,826	0,432	5,55112E-17	0,296
800	0,283	0,311	0,173	0,827	0,428	1,11022E-16	0,296
900	0,284	0,311	0,172	0,828	0,426	5,55112E-17	0,297
1000	0,286	0,311	0,171	0,829	0,424	5,55112E-17	0,298
1100	0,279	0,310	0,171	0,829	0,423	5,55112E-17	0,294
1200	0,280	0,311	0,170	0,830	0,421	5,55112E-17	0,294
1300	0,285	0,311	0,170	0,830	0,419	0	0,297
1400	0,292	0,311	0,169	0,831	0,418	5,55112E-17	0,301
1500	0,297	0,310	0,169	0,831	0,417	1,11022E-16	0,303
1600	0,263	0,310	0,168	0,832	0,416	1,11022E-16	0,285
1700	0,264	0,310	0,167	0,833	0,415	5,55112E-17	0,285
1800	0,263	0,310	0,167	0,833	0,414	0	0,285
1900	0,265	0,310	0,167	0,833	0,413	5,55112E-17	0,286
2000	0,263	0,309	0,166	0,834	0,413	5,55112E-17	0,284
2100	0,259	0,309	0,166	0,834	0,413	5,55112E-17	0,282
2200	0,259	0,309	0,166	0,834	0,412	0	0,282
2300	0,260	0,309	0,166	0,834	0,411	0	0,282
2400	0,259	0,308	0,165	0,835	0,410	5,55112E-17	0,281
2500	0,262	0,308	0,165	0,835	0,409	0	0,283
2600	0,259	0,308	0,164	0,836	0,408	0	0,282
2700	0,253	0,308	0,164	0,836	0,408	0	0,278
2800	0,248	0,307	0,164	0,836	0,408	0	0,274
2900	0,250	0,308	0,164	0,836	0,407	5,55112E-17	0,276
3000	0,250	0,308	0,163	0,837	0,407	5,55112E-17	0,276
3100	0,254	0,308	0,163	0,837	0,406	1,11022E-16	0,278
3200	0,245	0,307	0,163	0,837	0,405	1,11022E-16	0,273
3300	0,242	0,307	0,163	0,837	0,405	1,11022E-16	0,271
3400	0,244	0,307	0,163	0,837	0,405	5,55112E-17	0,272
3500	0,247	0,307	0,162	0,838	0,405	5,55112E-17	0,274
3600	0,250	0,307	0,162	0,838	0,404	5,55112E-17	0,276
3700	0,250	0,307	0,162	0,838	0,403	0	0,276
3800	0,251	0,307	0,162	0,838	0,403	0	0,276
3900	0,238	0,307	0,162	0,838	0,403	0	0,268
4000	0,239	0,307	0,162	0,838	0,403	0	0,269

Tabela B.2: Resultados do teste ST02Test

ST03Test

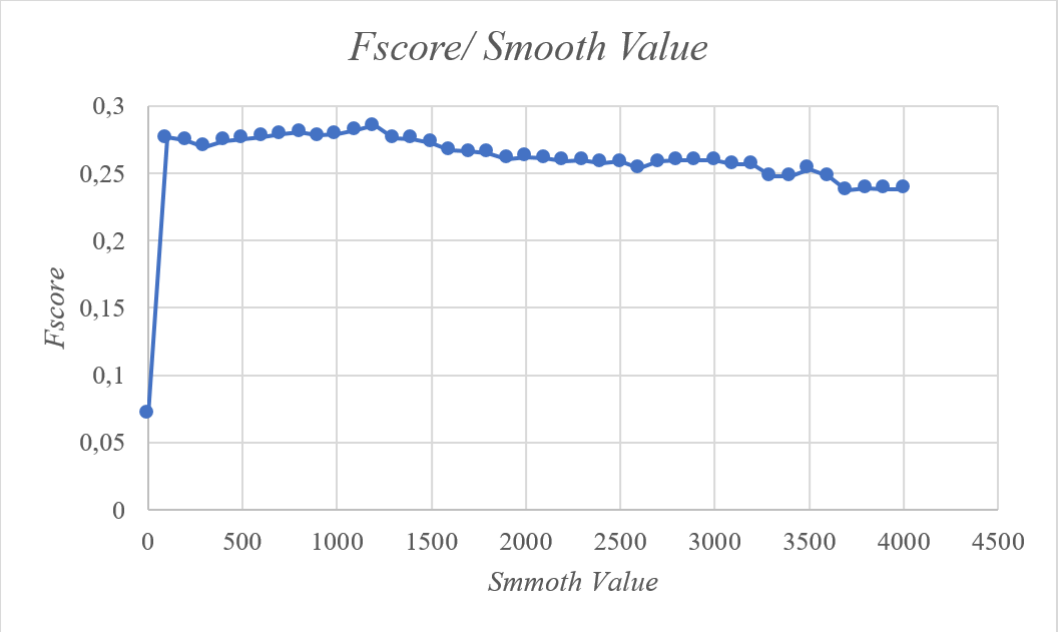

Figura B.7: ST03Test - Gráfico *Fscore/ Smooth Value*

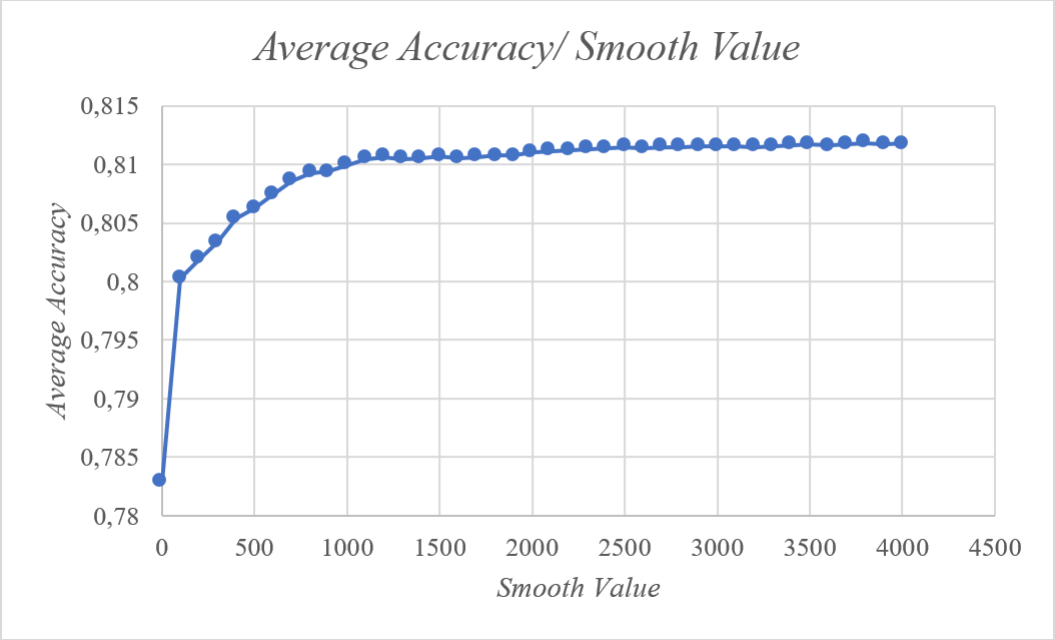

Figura B.8: ST03Test - Gráfico *Average Accuracy/ Smooth Value*

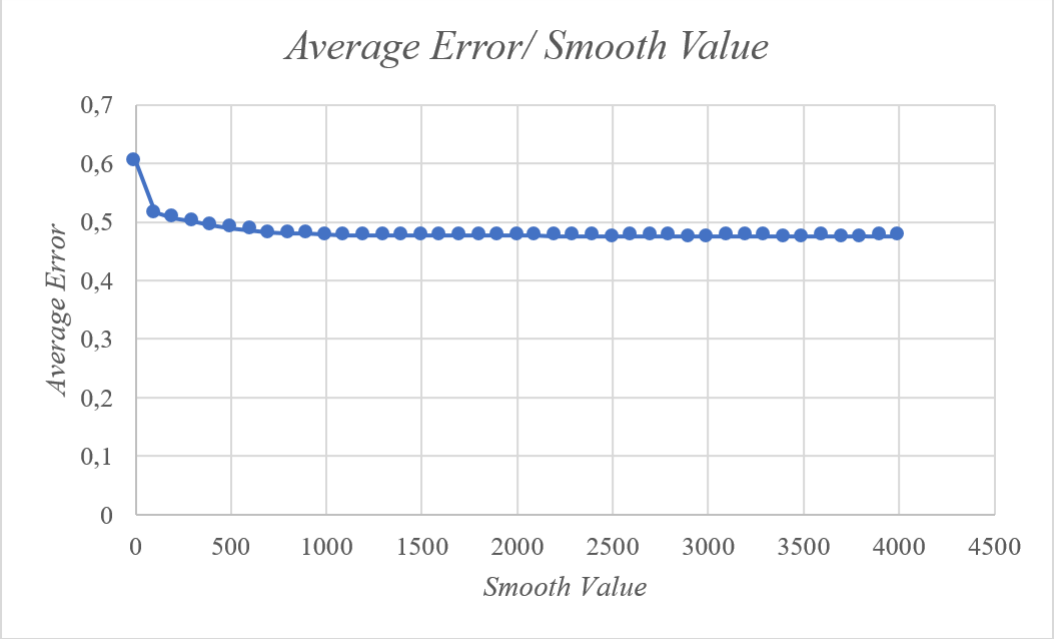

Figura B.9: ST03Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,050	0,125	0,217	0,783	0,603	1,11022E-16	0,071
100	0,265	0,290	0,200	0,800	0,516	1,11022E-16	0,277
200	0,269	0,281	0,198	0,802	0,506	0	0,275
300	0,264	0,275	0,197	0,803	0,500	1,11022E-16	0,269
400	0,269	0,278	0,195	0,805	0,494	5,55112E-17	0,274
500	0,272	0,279	0,194	0,806	0,490	5,55112E-17	0,276
600	0,273	0,281	0,193	0,807	0,486	1,11022E-16	0,277
700	0,276	0,283	0,191	0,809	0,482	5,55112E-17	0,279
800	0,278	0,283	0,191	0,809	0,480	1,11022E-16	0,280
900	0,274	0,282	0,191	0,809	0,480	1,11022E-16	0,278
1000	0,276	0,282	0,190	0,810	0,478	0	0,279
1100	0,282	0,282	0,190	0,810	0,477	0	0,282
1200	0,289	0,282	0,189	0,811	0,477	5,55112E-17	0,285
1300	0,271	0,281	0,190	0,810	0,477	0	0,276
1400	0,271	0,280	0,189	0,811	0,477	5,55112E-17	0,276
1500	0,265	0,280	0,189	0,811	0,477	5,55112E-17	0,272
1600	0,256	0,279	0,190	0,811	0,477	0	0,267
1700	0,255	0,278	0,189	0,811	0,477	0	0,266
1800	0,253	0,278	0,189	0,811	0,477	0	0,265
1900	0,245	0,278	0,189	0,811	0,477	0	0,260
2000	0,248	0,278	0,189	0,811	0,477	5,55112E-17	0,262
2100	0,247	0,278	0,189	0,811	0,476	1,11022E-16	0,261
2200	0,243	0,277	0,189	0,811	0,476	1,11022E-16	0,259
2300	0,244	0,277	0,189	0,811	0,476	5,55112E-17	0,260
2400	0,240	0,277	0,189	0,811	0,476	5,55112E-17	0,257
2500	0,243	0,277	0,189	0,811	0,476	0	0,259
2600	0,235	0,276	0,189	0,811	0,476	0	0,254
2700	0,243	0,276	0,188	0,812	0,476	5,55112E-17	0,259
2800	0,245	0,276	0,189	0,811	0,476	5,55112E-17	0,259
2900	0,246	0,276	0,188	0,812	0,476	0	0,260
3000	0,246	0,276	0,188	0,812	0,476	0	0,260
3100	0,240	0,276	0,188	0,812	0,476	5,55112E-17	0,257
3200	0,240	0,275	0,189	0,811	0,476	0	0,256
3300	0,226	0,275	0,188	0,812	0,476	5,55112E-17	0,248
3400	0,226	0,275	0,188	0,812	0,476	0	0,248
3500	0,234	0,275	0,188	0,812	0,476	0	0,253
3600	0,226	0,275	0,188	0,812	0,476	0	0,248
3700	0,209	0,275	0,188	0,812	0,476	0	0,238
3800	0,211	0,275	0,188	0,812	0,475	5,55112E-17	0,239
3900	0,211	0,274	0,188	0,812	0,476	5,55112E-17	0,238
4000	0,211	0,274	0,188	0,812	0,476	5,55112E-17	0,238

Tabela B.3: Resultados do teste ST03Test

ST04Test

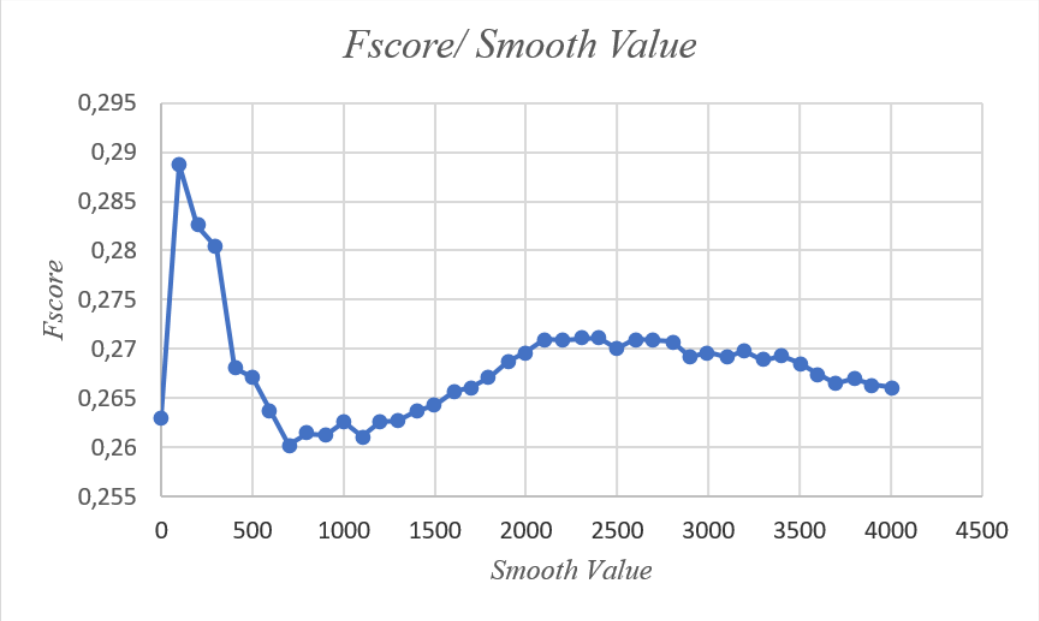

Figura B.10: ST04Test - Gráfico *Fscore/ Smooth Value*

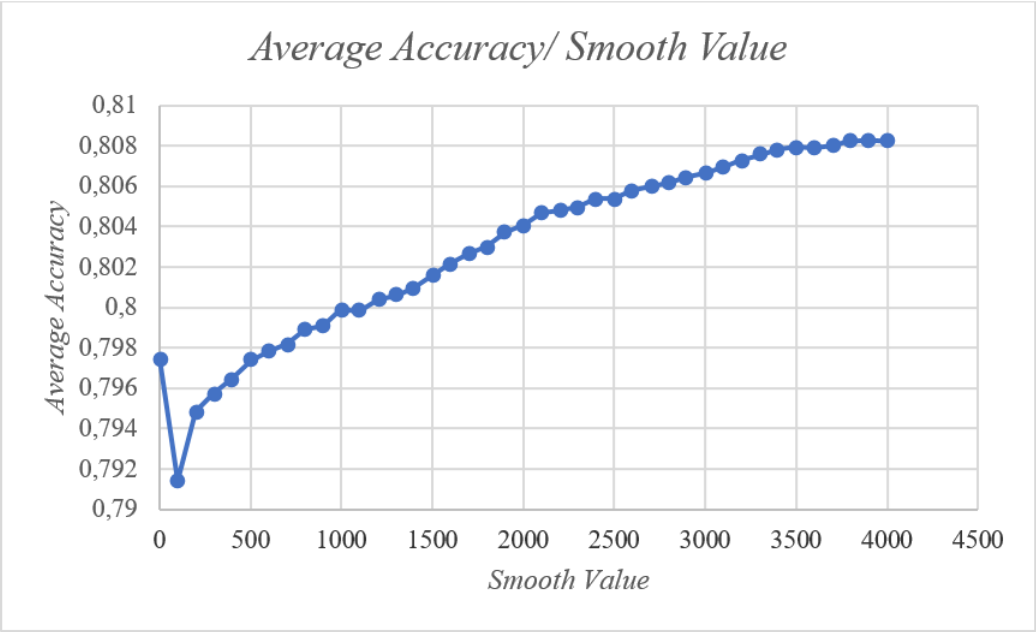

Figura B.11: ST04Test - Gráfico *Average Accuracy/ Smooth Value*

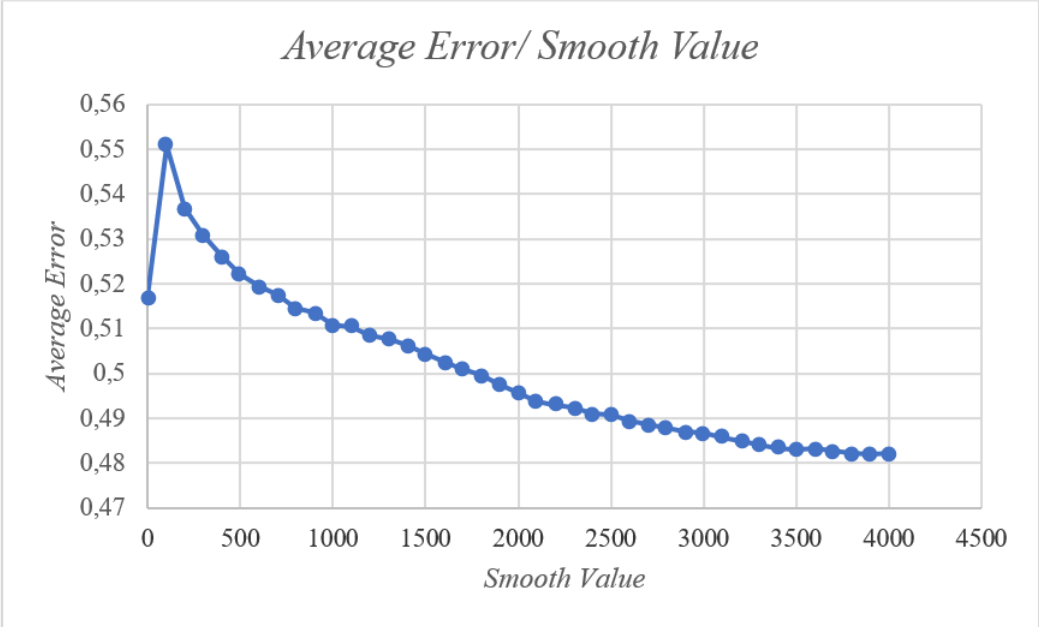

Figura B.12: ST04Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,257	0,270	0,203	0,797	0,517	1,11022E-16	0,263
100	0,264	0,318	0,209	0,791	0,551	1,11022E-16	0,289
200	0,264	0,304	0,205	0,795	0,537	0	0,283
300	0,261	0,303	0,204	0,796	0,531	1,11022E-16	0,280
400	0,256	0,281	0,204	0,796	0,526	0	0,268
500	0,256	0,279	0,203	0,797	0,522	1,11022E-16	0,267
600	0,254	0,274	0,202	0,798	0,519	0	0,264
700	0,253	0,267	0,202	0,798	0,518	1,11022E-16	0,260
800	0,254	0,269	0,201	0,799	0,515	1,11022E-16	0,261
900	0,254	0,269	0,201	0,799	0,514	1,11022E-16	0,261
1000	0,255	0,271	0,200	0,800	0,511	1,11022E-16	0,263
1100	0,253	0,269	0,200	0,800	0,510	1,11022E-16	0,261
1200	0,255	0,270	0,200	0,800	0,508	1,11022E-16	0,263
1300	0,255	0,270	0,199	0,801	0,508	1,11022E-16	0,263
1400	0,256	0,271	0,199	0,801	0,506	0	0,264
1500	0,257	0,272	0,198	0,802	0,504	1,11022E-16	0,264
1600	0,259	0,273	0,198	0,802	0,502	1,11022E-16	0,266
1700	0,259	0,274	0,197	0,803	0,501	0	0,266
1800	0,260	0,275	0,197	0,803	0,500	0	0,267
1900	0,261	0,276	0,196	0,804	0,497	0	0,269
2000	0,262	0,277	0,196	0,804	0,496	5,55112E-17	0,270
2100	0,264	0,278	0,195	0,805	0,494	5,55112E-17	0,271
2200	0,264	0,278	0,195	0,805	0,493	0	0,271
2300	0,264	0,279	0,195	0,805	0,492	0	0,271
2400	0,264	0,279	0,195	0,805	0,491	0	0,271
2500	0,262	0,278	0,195	0,805	0,491	5,55112E-17	0,270
2600	0,263	0,279	0,194	0,806	0,489	0	0,271
2700	0,263	0,279	0,194	0,806	0,489	1,11022E-16	0,271
2800	0,263	0,279	0,194	0,806	0,488	5,55112E-17	0,271
2900	0,260	0,279	0,194	0,806	0,487	1,11022E-16	0,269
3000	0,261	0,280	0,193	0,807	0,487	5,55112E-17	0,270
3100	0,259	0,280	0,193	0,807	0,486	0	0,269
3200	0,260	0,280	0,193	0,807	0,485	0	0,270
3300	0,258	0,280	0,192	0,808	0,484	0	0,269
3400	0,259	0,281	0,192	0,808	0,483	0	0,269
3500	0,258	0,281	0,192	0,808	0,483	0	0,269
3600	0,256	0,280	0,192	0,808	0,483	0	0,267
3700	0,254	0,280	0,192	0,808	0,483	0	0,266
3800	0,255	0,281	0,192	0,808	0,482	5,55112E-17	0,267
3900	0,254	0,280	0,192	0,808	0,482	5,55112E-17	0,266
4000	0,253	0,280	0,192	0,808	0,482	0	0,266

Tabela B.4: Resultados do teste ST04Test

ST05Test

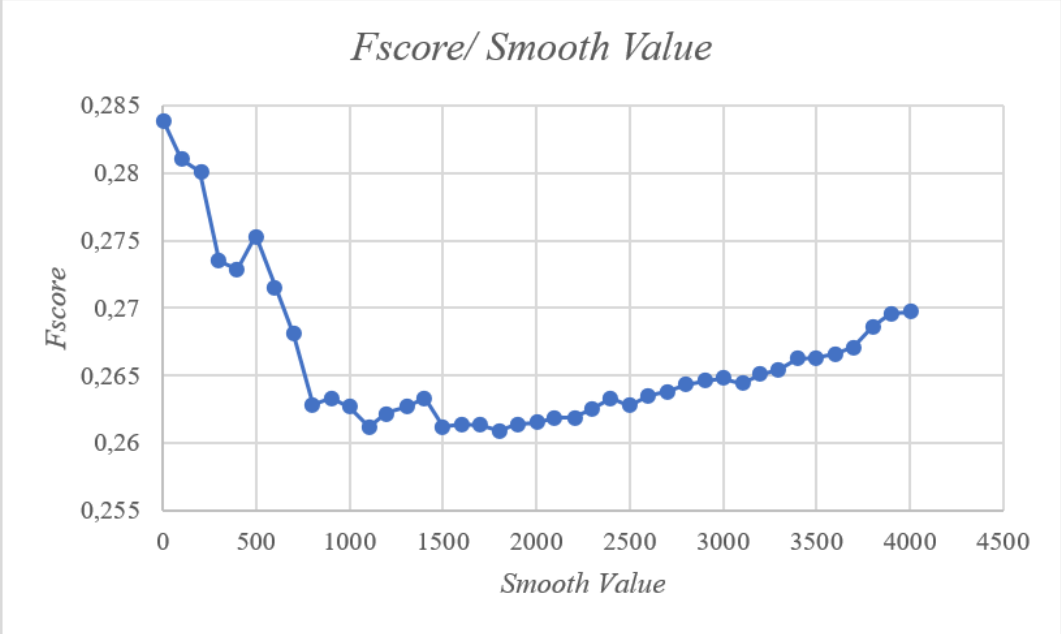

Figura B.13: ST05Test - Gráfico *Fscore/ Smooth Value*

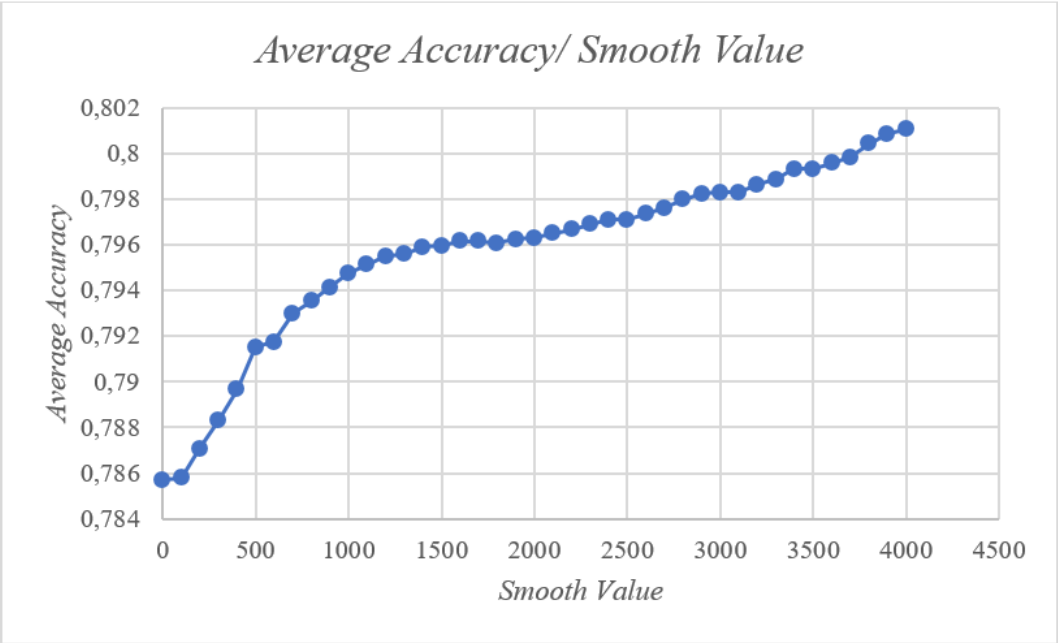

Figura B.14: ST05Test - Gráfico *Average Accuracy/ Smooth Value*

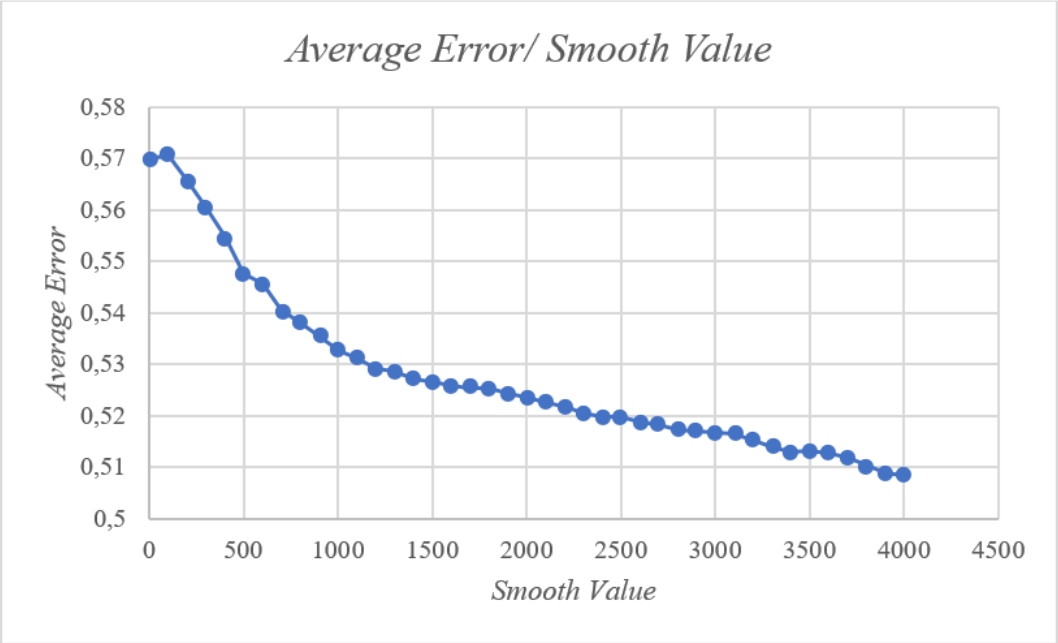

Figura B.15: ST05Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,262	0,310	0,214	0,786	0,570	1,11022E-16	0,284
100	0,260	0,305	0,214	0,786	0,571	1,11022E-16	0,281
200	0,260	0,304	0,213	0,787	0,566	1,11022E-16	0,280
300	0,258	0,291	0,212	0,788	0,561	1,11022E-16	0,273
400	0,258	0,290	0,210	0,790	0,555	0	0,273
500	0,259	0,294	0,208	0,792	0,548	0	0,275
600	0,258	0,287	0,208	0,792	0,546	1,11022E-16	0,272
700	0,259	0,278	0,207	0,793	0,540	1,11022E-16	0,268
800	0,257	0,269	0,206	0,794	0,538	1,11022E-16	0,263
900	0,257	0,270	0,206	0,794	0,535	1,11022E-16	0,263
1000	0,258	0,268	0,205	0,795	0,533	1,11022E-16	0,263
1100	0,258	0,265	0,205	0,795	0,531	1,11022E-16	0,261
1200	0,259	0,266	0,204	0,796	0,529	1,11022E-16	0,262
1300	0,259	0,266	0,204	0,796	0,529	1,11022E-16	0,263
1400	0,260	0,267	0,204	0,796	0,527	1,11022E-16	0,263
1500	0,259	0,263	0,204	0,796	0,527	1,11022E-16	0,261
1600	0,259	0,264	0,204	0,796	0,526	1,11022E-16	0,261
1700	0,259	0,264	0,204	0,796	0,525	1,11022E-16	0,261
1800	0,258	0,264	0,204	0,796	0,525	1,11022E-16	0,261
1900	0,259	0,264	0,204	0,796	0,524	1,11022E-16	0,261
2000	0,259	0,264	0,204	0,796	0,524	1,11022E-16	0,262
2100	0,259	0,264	0,204	0,796	0,523	0	0,262
2200	0,259	0,265	0,203	0,797	0,522	1,11022E-16	0,262
2300	0,260	0,265	0,203	0,797	0,521	0	0,263
2400	0,261	0,266	0,203	0,797	0,520	1,11022E-16	0,263
2500	0,260	0,266	0,203	0,797	0,520	1,11022E-16	0,263
2600	0,261	0,266	0,203	0,797	0,519	0	0,264
2700	0,261	0,267	0,202	0,798	0,518	0	0,264
2800	0,261	0,268	0,202	0,798	0,517	1,11022E-16	0,264
2900	0,262	0,268	0,202	0,798	0,517	0	0,265
3000	0,262	0,268	0,202	0,798	0,517	1,11022E-16	0,265
3100	0,261	0,268	0,202	0,798	0,516	1,11022E-16	0,264
3200	0,262	0,268	0,201	0,799	0,515	1,11022E-16	0,265
3300	0,262	0,269	0,201	0,799	0,514	0	0,265
3400	0,263	0,270	0,201	0,799	0,513	1,11022E-16	0,266
3500	0,263	0,270	0,201	0,799	0,513	0	0,266
3600	0,264	0,270	0,200	0,800	0,513	1,11022E-16	0,267
3700	0,264	0,270	0,200	0,800	0,512	0	0,267
3800	0,266	0,271	0,200	0,800	0,510	1,11022E-16	0,269
3900	0,267	0,272	0,199	0,801	0,509	1,11022E-16	0,270
4000	0,267	0,272	0,199	0,801	0,508	1,11022E-16	0,270

Tabela B.5: Resultados do teste ST05Test

ST06Test

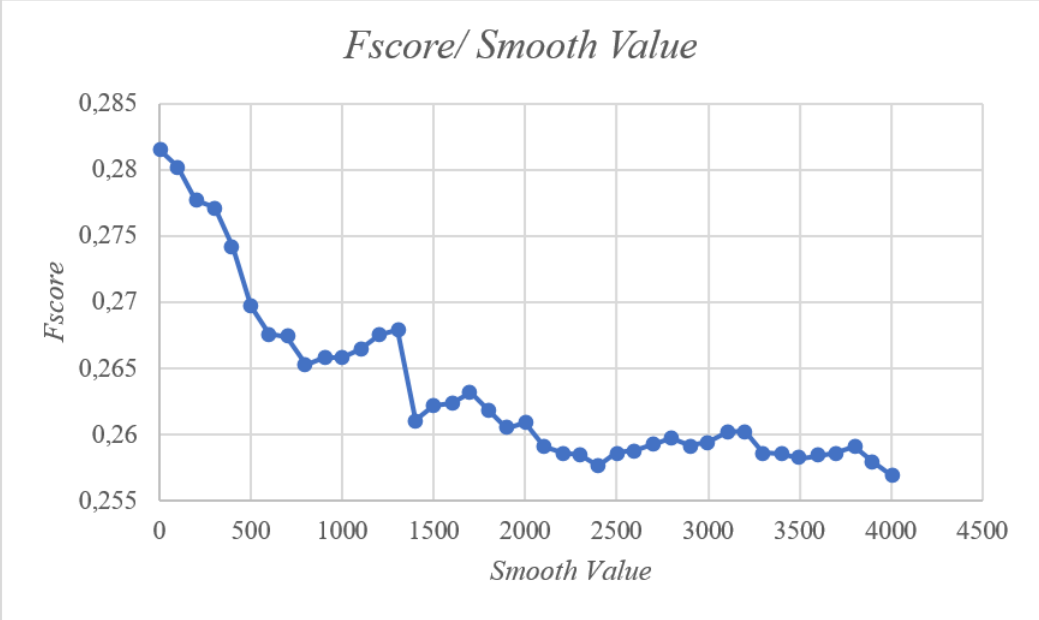

Figura B.16: ST06Test - Gráfico *Fscore/ Smooth Value*

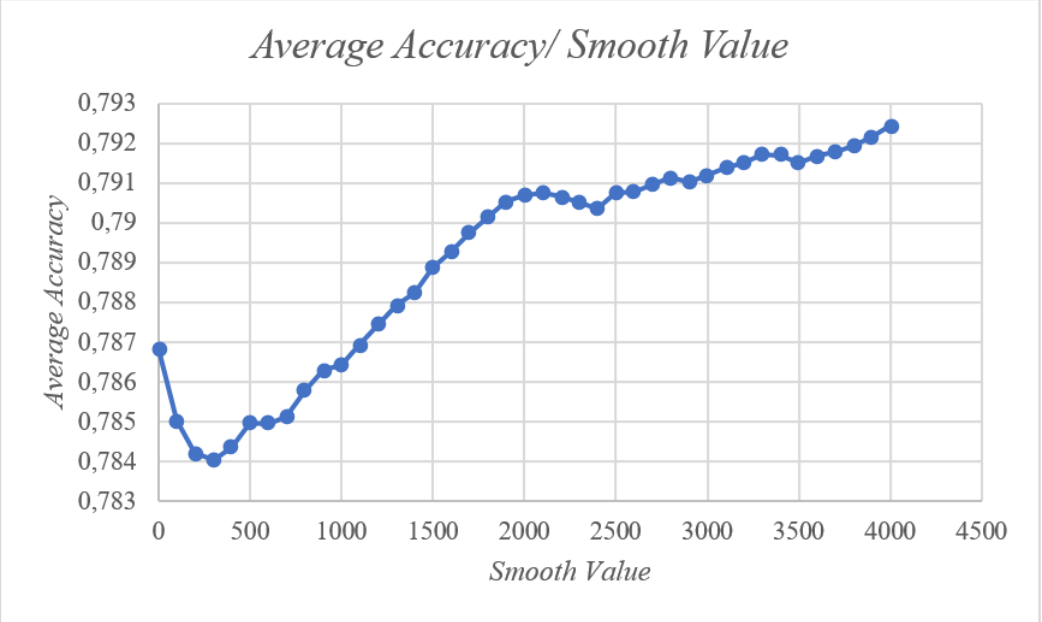

Figura B.17: ST06Test - Gráfico *Average Accuracy/ Smooth Value*

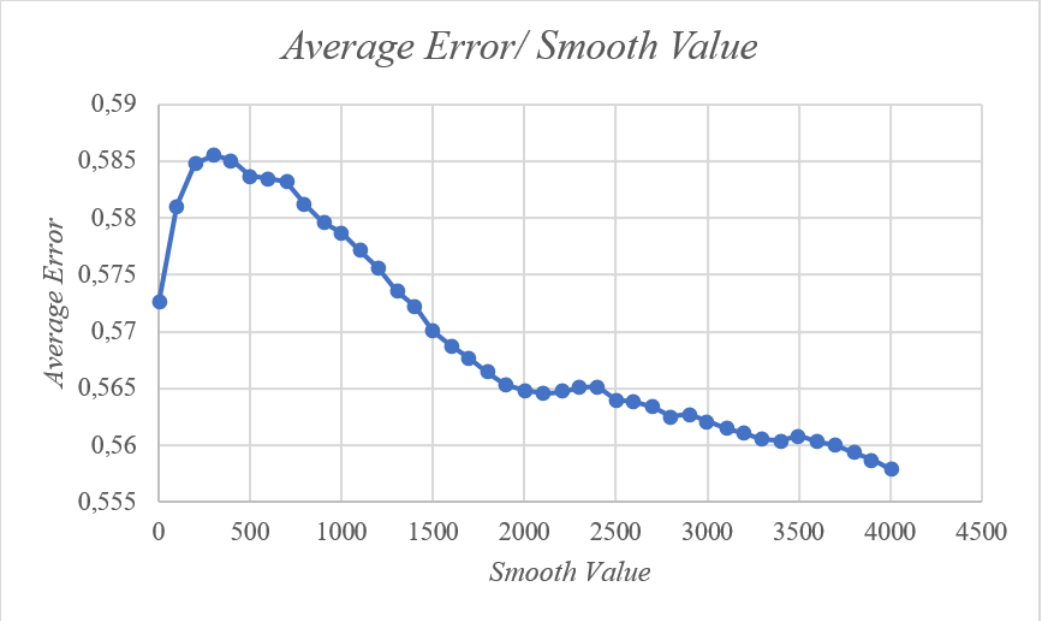

Figura B.18: ST06Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,260	0,307	0,213	0,787	0,573	1,11022E-16	0,282
100	0,261	0,302	0,215	0,785	0,581	1,11022E-16	0,280
200	0,262	0,296	0,216	0,784	0,585	1,11022E-16	0,278
300	0,262	0,295	0,216	0,784	0,586	0	0,277
400	0,262	0,288	0,216	0,784	0,585	0	0,274
500	0,262	0,277	0,215	0,785	0,584	1,11022E-16	0,270
600	0,262	0,274	0,215	0,785	0,583	1,11022E-16	0,268
700	0,261	0,274	0,215	0,785	0,583	1,11022E-16	0,267
800	0,262	0,268	0,214	0,786	0,581	1,11022E-16	0,265
900	0,263	0,269	0,214	0,786	0,580	1,11022E-16	0,266
1000	0,263	0,269	0,214	0,786	0,579	1,11022E-16	0,266
1100	0,263	0,270	0,213	0,787	0,577	0	0,266
1200	0,264	0,271	0,213	0,787	0,576	1,11022E-16	0,268
1300	0,264	0,272	0,212	0,788	0,574	1,11022E-16	0,268
1400	0,264	0,258	0,212	0,788	0,572	1,11022E-16	0,261
1500	0,265	0,260	0,211	0,789	0,570	0	0,262
1600	0,265	0,260	0,211	0,789	0,569	0	0,262
1700	0,266	0,261	0,210	0,790	0,568	1,11022E-16	0,263
1800	0,266	0,258	0,210	0,790	0,566	0	0,262
1900	0,266	0,256	0,209	0,791	0,565	1,11022E-16	0,261
2000	0,266	0,256	0,209	0,791	0,565	1,11022E-16	0,261
2100	0,266	0,253	0,209	0,791	0,565	0	0,259
2200	0,265	0,253	0,209	0,791	0,565	0	0,259
2300	0,265	0,252	0,209	0,791	0,565	1,11022E-16	0,259
2400	0,263	0,252	0,210	0,790	0,565	1,11022E-16	0,258
2500	0,265	0,253	0,209	0,791	0,564	0	0,259
2600	0,265	0,253	0,209	0,791	0,564	0	0,259
2700	0,265	0,254	0,209	0,791	0,563	0	0,259
2800	0,266	0,254	0,209	0,791	0,563	0	0,260
2900	0,265	0,254	0,209	0,791	0,563	1,11022E-16	0,259
3000	0,265	0,254	0,209	0,791	0,562	1,11022E-16	0,259
3100	0,266	0,255	0,209	0,791	0,561	0	0,260
3200	0,266	0,255	0,208	0,792	0,561	0	0,260
3300	0,266	0,252	0,208	0,792	0,561	1,11022E-16	0,259
3400	0,266	0,252	0,208	0,792	0,560	0	0,259
3500	0,266	0,251	0,208	0,792	0,561	0	0,258
3600	0,266	0,252	0,208	0,792	0,560	1,11022E-16	0,258
3700	0,266	0,252	0,208	0,792	0,560	0	0,259
3800	0,267	0,252	0,208	0,792	0,559	0	0,259
3900	0,267	0,249	0,208	0,792	0,559	0	0,258
4000	0,268	0,246	0,208	0,792	0,558	0	0,257

Tabela B.6: Resultados do teste ST06Test

ST07Test

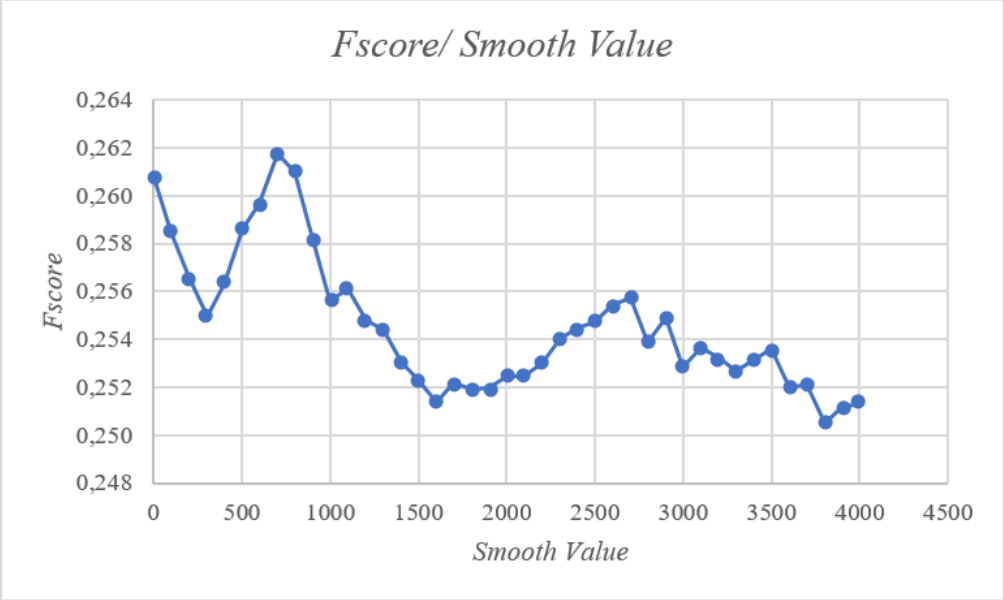

Figura B.19: ST07Test - Gráfico *Fscore/ Smooth Value*

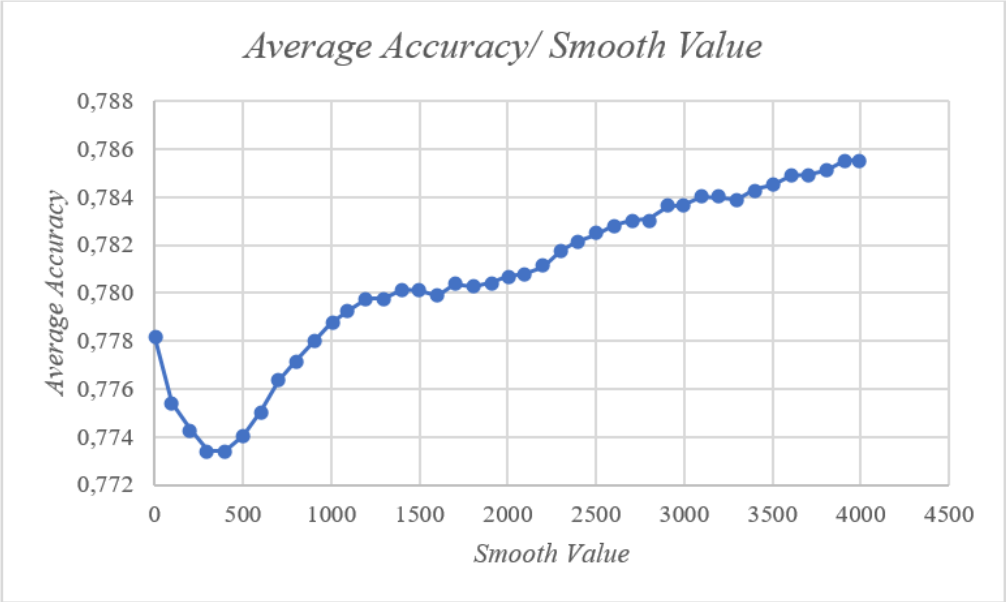

Figura B.20: ST07Test - Gráfico *Average Accuracy/ Smooth Value*

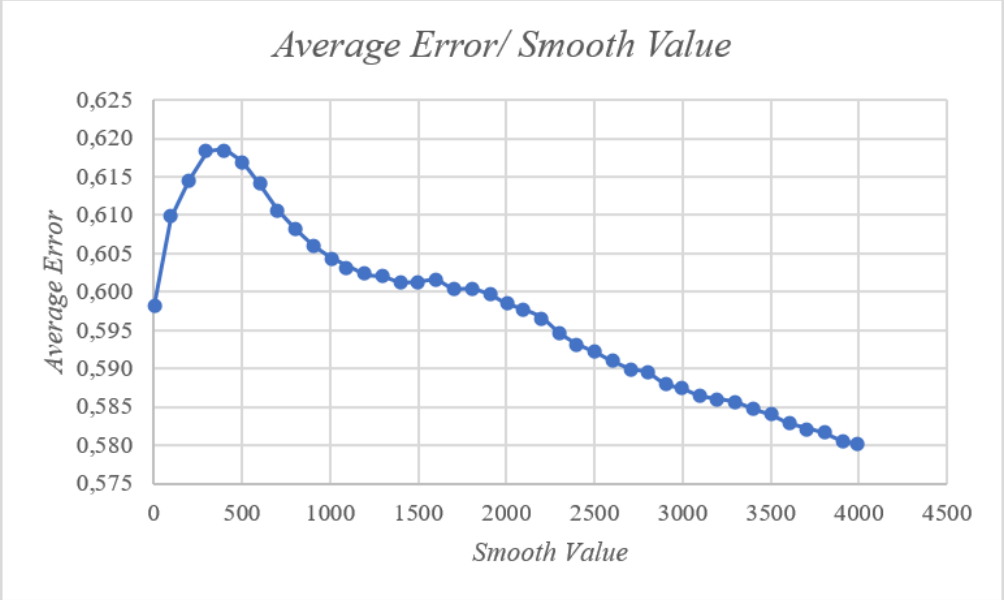

Figura B.21: ST07Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,254	0,268	0,222	0,778	0,598	0	0,261
100	0,253	0,264	0,225	0,775	0,610	1,11022E-16	0,259
200	0,253	0,261	0,226	0,774	0,615	0	0,257
300	0,252	0,258	0,227	0,773	0,618	1,11022E-16	0,255
400	0,253	0,260	0,227	0,773	0,619	1,11022E-16	0,256
500	0,254	0,264	0,226	0,774	0,617	1,11022E-16	0,259
600	0,255	0,265	0,225	0,775	0,614	0	0,260
700	0,257	0,267	0,224	0,776	0,611	0	0,262
800	0,258	0,264	0,223	0,777	0,608	1,11022E-16	0,261
900	0,258	0,258	0,222	0,778	0,606	1,11022E-16	0,258
1000	0,259	0,252	0,221	0,779	0,605	0	0,256
1100	0,260	0,253	0,221	0,779	0,603	0	0,256
1200	0,260	0,250	0,220	0,780	0,602	1,11022E-16	0,255
1300	0,259	0,250	0,220	0,780	0,602	1,11022E-16	0,254
1400	0,259	0,247	0,220	0,780	0,601	0	0,253
1500	0,258	0,247	0,220	0,780	0,601	1,11022E-16	0,252
1600	0,257	0,246	0,220	0,780	0,602	1,11022E-16	0,251
1700	0,258	0,247	0,220	0,780	0,600	1,11022E-16	0,252
1800	0,257	0,247	0,220	0,780	0,600	0	0,252
1900	0,257	0,247	0,220	0,780	0,600	1,11022E-16	0,252
2000	0,257	0,248	0,219	0,781	0,598	1,11022E-16	0,252
2100	0,257	0,248	0,219	0,781	0,598	1,11022E-16	0,252
2200	0,258	0,248	0,219	0,781	0,597	1,11022E-16	0,253
2300	0,258	0,250	0,218	0,782	0,594	1,11022E-16	0,254
2400	0,258	0,251	0,218	0,782	0,593	1,11022E-16	0,254
2500	0,258	0,251	0,218	0,782	0,592	0	0,255
2600	0,259	0,252	0,217	0,783	0,591	1,11022E-16	0,255
2700	0,259	0,252	0,217	0,783	0,590	1,11022E-16	0,256
2800	0,259	0,249	0,217	0,783	0,590	1,11022E-16	0,254
2900	0,260	0,250	0,216	0,784	0,588	1,11022E-16	0,255
3000	0,259	0,247	0,216	0,784	0,587	1,11022E-16	0,253
3100	0,260	0,248	0,216	0,784	0,586	0	0,254
3200	0,259	0,248	0,216	0,784	0,586	0	0,253
3300	0,258	0,248	0,216	0,784	0,586	0	0,253
3400	0,258	0,248	0,216	0,784	0,585	1,11022E-16	0,253
3500	0,258	0,249	0,215	0,785	0,584	1,11022E-16	0,254
3600	0,258	0,246	0,215	0,785	0,583	0	0,252
3700	0,258	0,246	0,215	0,785	0,582	1,11022E-16	0,252
3800	0,258	0,243	0,215	0,785	0,582	1,11022E-16	0,250
3900	0,259	0,244	0,215	0,785	0,581	1,11022E-16	0,251
4000	0,259	0,244	0,214	0,786	0,580	0	0,251

Tabela B.7: Resultados do teste ST07Test

ST08Test

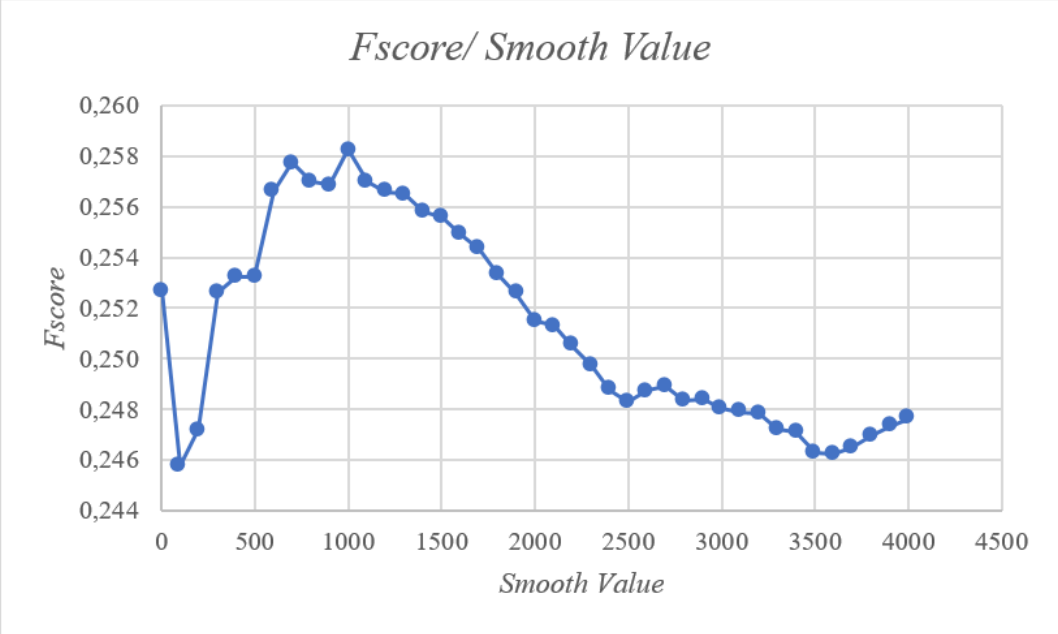

Figura B.22: ST08Test - Gráfico *Fscore/ Smooth Value*

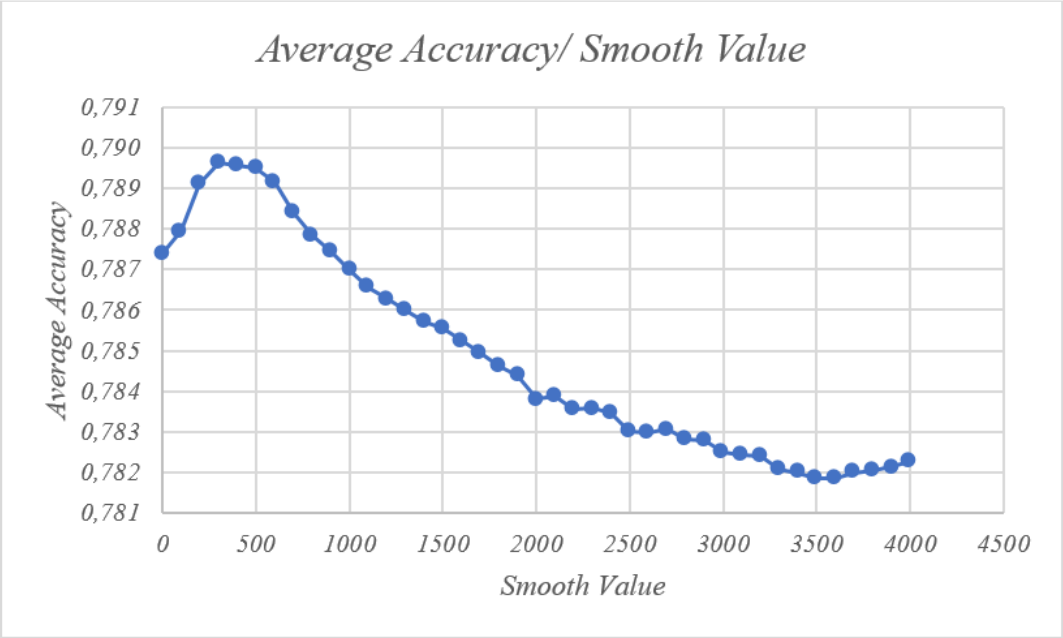

Figura B.23: ST08Test - Gráfico *Average Accuracy/ Smooth Value*

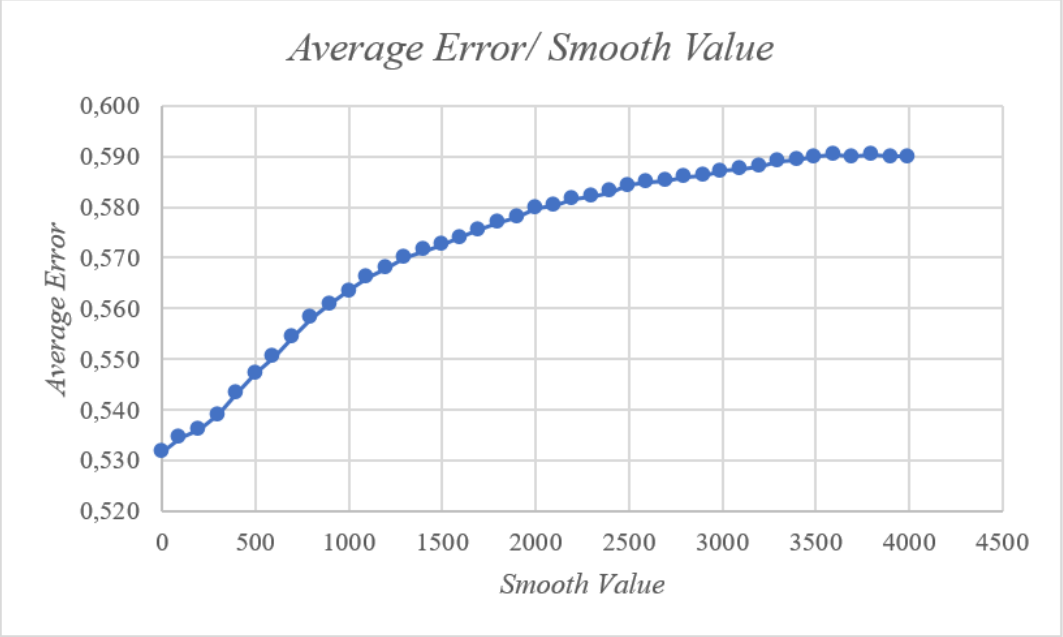

Figura B.24: ST08Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,242	0,265	0,213	0,787	0,532	1,11022E-16	0,253
100	0,230	0,264	0,212	0,788	0,535	0	0,246
200	0,233	0,263	0,211	0,789	0,536	1,11022E-16	0,247
300	0,236	0,271	0,210	0,790	0,539	1,11022E-16	0,253
400	0,239	0,269	0,210	0,790	0,543	0	0,253
500	0,241	0,266	0,211	0,789	0,547	1,11022E-16	0,253
600	0,242	0,273	0,211	0,789	0,551	0	0,257
700	0,242	0,275	0,212	0,788	0,554	0	0,258
800	0,243	0,272	0,212	0,788	0,558	1,11022E-16	0,257
900	0,245	0,270	0,213	0,787	0,561	1,11022E-16	0,257
1000	0,245	0,273	0,213	0,787	0,563	0	0,258
1100	0,244	0,271	0,213	0,787	0,566	0	0,257
1200	0,245	0,270	0,214	0,786	0,568	0	0,257
1300	0,246	0,268	0,214	0,786	0,570	1,11022E-16	0,256
1400	0,246	0,267	0,214	0,786	0,571	1,11022E-16	0,256
1500	0,246	0,266	0,214	0,786	0,573	1,11022E-16	0,256
1600	0,246	0,265	0,215	0,785	0,574	0	0,255
1700	0,246	0,264	0,215	0,785	0,576	1,11022E-16	0,254
1800	0,245	0,263	0,215	0,785	0,577	1,11022E-16	0,253
1900	0,244	0,262	0,216	0,784	0,578	1,11022E-16	0,253
2000	0,243	0,260	0,216	0,784	0,580	1,11022E-16	0,251
2100	0,243	0,260	0,216	0,784	0,580	0	0,251
2200	0,243	0,259	0,216	0,784	0,581	0	0,251
2300	0,242	0,258	0,216	0,784	0,582	0	0,250
2400	0,241	0,257	0,217	0,783	0,583	1,11022E-16	0,249
2500	0,241	0,256	0,217	0,783	0,584	1,11022E-16	0,248
2600	0,242	0,256	0,217	0,783	0,585	1,11022E-16	0,249
2700	0,242	0,256	0,217	0,783	0,585	0	0,249
2800	0,242	0,255	0,217	0,783	0,586	0	0,248
2900	0,242	0,255	0,217	0,783	0,586	1,11022E-16	0,248
3000	0,242	0,254	0,217	0,783	0,587	0	0,248
3100	0,242	0,254	0,218	0,782	0,587	1,11022E-16	0,248
3200	0,242	0,254	0,218	0,782	0,588	1,11022E-16	0,248
3300	0,242	0,253	0,218	0,782	0,589	0	0,247
3400	0,242	0,253	0,218	0,782	0,589	1,11022E-16	0,247
3500	0,241	0,252	0,218	0,782	0,590	1,11022E-16	0,246
3600	0,241	0,252	0,218	0,782	0,590	0	0,246
3700	0,241	0,252	0,218	0,782	0,590	0	0,246
3800	0,242	0,252	0,218	0,782	0,590	0	0,247
3900	0,243	0,252	0,218	0,782	0,590	0	0,247
4000	0,244	0,252	0,218	0,782	0,590	1,11022E-16	0,248

Tabela B.8: Resultados do teste ST08Test

ST09Test

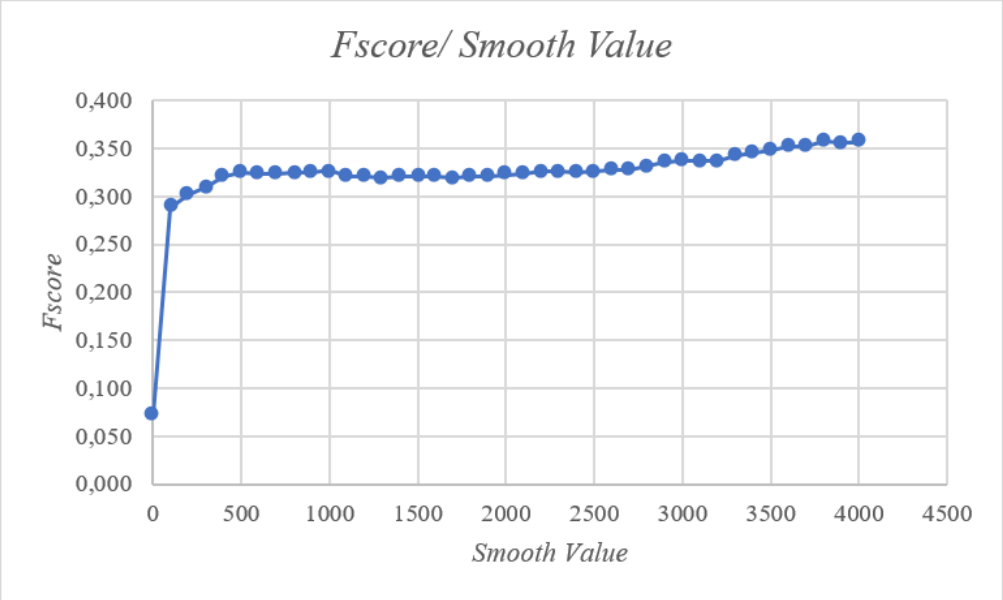

Figura B.25: ST09Test - Gráfico *Fscore/ Smooth Value*

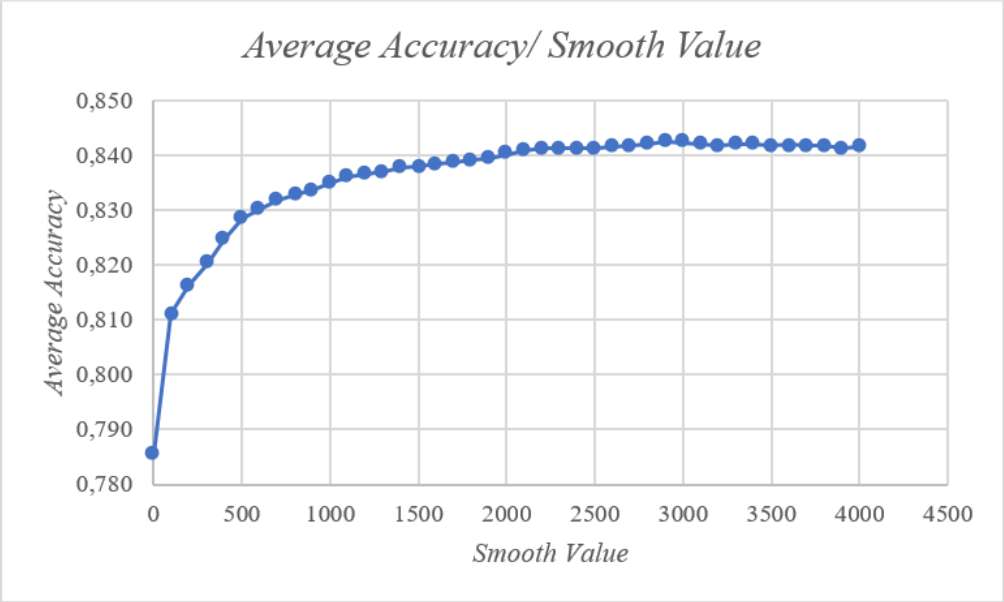

Figura B.26: ST09Test - Gráfico *Average Accuracy/ Smooth Value*

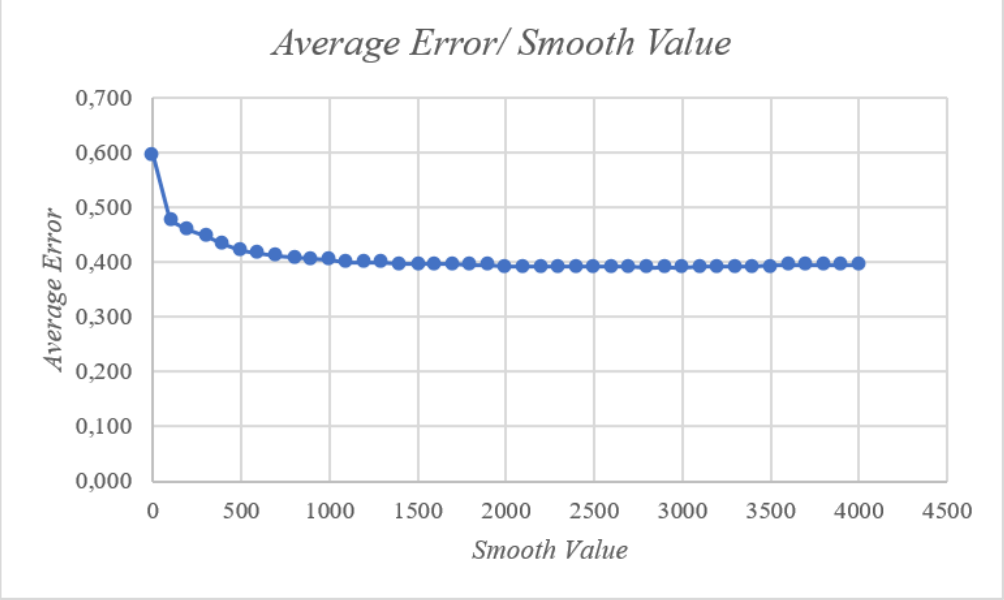

Figura B.27: ST09Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,051	0,125	0,215	0,785	0,595	0,051	0,125
100	0,282	0,295	0,189	0,811	0,476	0,282	0,295
200	0,295	0,307	0,184	0,816	0,459	0,295	0,307
300	0,304	0,315	0,180	0,820	0,446	0,304	0,315
400	0,316	0,324	0,175	0,825	0,433	0,316	0,324
500	0,321	0,329	0,172	0,828	0,421	0,321	0,329
600	0,316	0,330	0,170	0,830	0,416	0,316	0,330
700	0,315	0,331	0,168	0,832	0,412	0,315	0,331
800	0,317	0,331	0,167	0,833	0,409	0,317	0,331
900	0,320	0,331	0,166	0,834	0,407	0,320	0,331
1000	0,323	0,332	0,165	0,835	0,403	0,323	0,332
1100	0,310	0,332	0,164	0,836	0,401	0,310	0,332
1200	0,310	0,332	0,163	0,837	0,399	0,310	0,332
1300	0,309	0,331	0,163	0,837	0,399	0,309	0,331
1400	0,311	0,330	0,162	0,838	0,397	0,311	0,330
1500	0,312	0,330	0,162	0,838	0,397	0,312	0,330
1600	0,313	0,328	0,162	0,838	0,396	0,313	0,328
1700	0,313	0,328	0,161	0,839	0,396	0,313	0,328
1800	0,314	0,327	0,161	0,839	0,395	0,314	0,327
1900	0,316	0,326	0,161	0,839	0,395	0,316	0,326
2000	0,319	0,327	0,160	0,840	0,393	0,319	0,327
2100	0,322	0,326	0,159	0,841	0,392	0,322	0,326
2200	0,326	0,326	0,159	0,841	0,391	0,326	0,326
2300	0,327	0,325	0,159	0,841	0,391	0,327	0,325
2400	0,328	0,323	0,159	0,841	0,392	0,328	0,323
2500	0,330	0,322	0,159	0,841	0,392	0,330	0,322
2600	0,334	0,321	0,158	0,842	0,391	0,334	0,321
2700	0,336	0,320	0,158	0,842	0,391	0,336	0,320
2800	0,345	0,320	0,158	0,842	0,391	0,345	0,320
2900	0,353	0,319	0,158	0,842	0,390	0,353	0,319
3000	0,359	0,318	0,158	0,842	0,391	0,359	0,318
3100	0,359	0,317	0,158	0,842	0,392	0,359	0,317
3200	0,362	0,315	0,158	0,842	0,393	0,362	0,315
3300	0,375	0,314	0,158	0,842	0,393	0,375	0,314
3400	0,384	0,313	0,158	0,842	0,393	0,384	0,313
3500	0,393	0,312	0,158	0,842	0,394	0,393	0,312
3600	0,406	0,311	0,158	0,842	0,394	0,406	0,311
3700	0,407	0,310	0,158	0,842	0,395	0,407	0,310
3800	0,425	0,309	0,158	0,842	0,395	0,425	0,309
3900	0,422	0,308	0,159	0,841	0,396	0,422	0,308
4000	0,424	0,308	0,158	0,842	0,396	0,424	0,308

Tabela B.9: Resultados do teste ST09Test

ST10Test

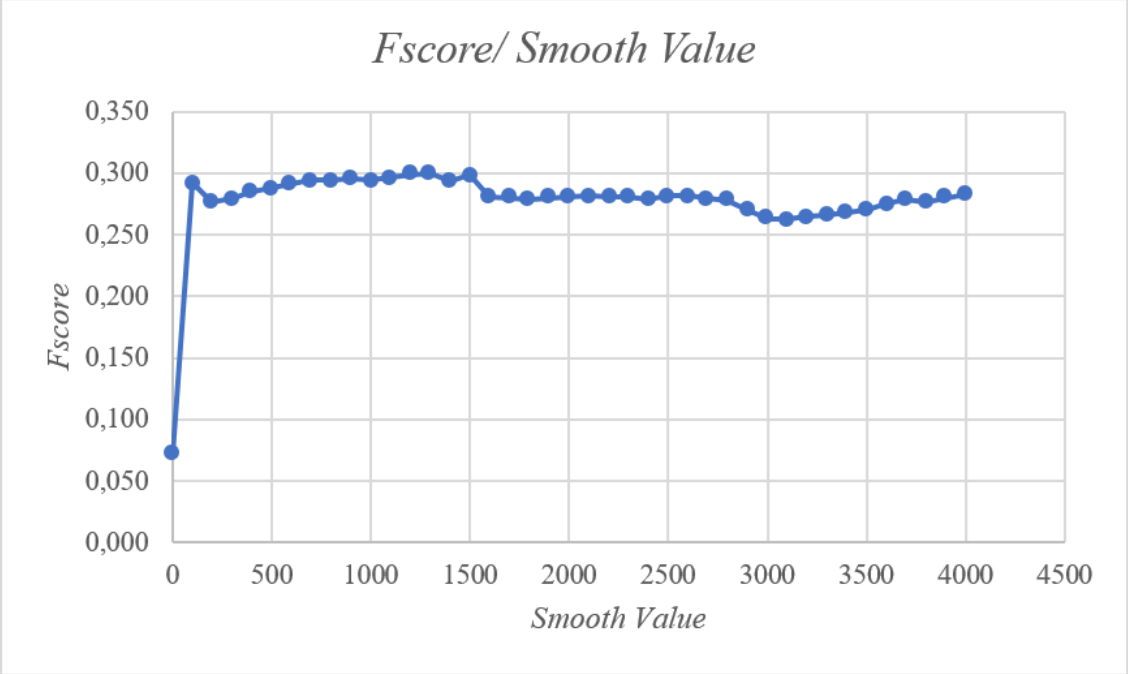

Figura B.28: ST10Test - Gráfico *Fscore/ Smooth Value*

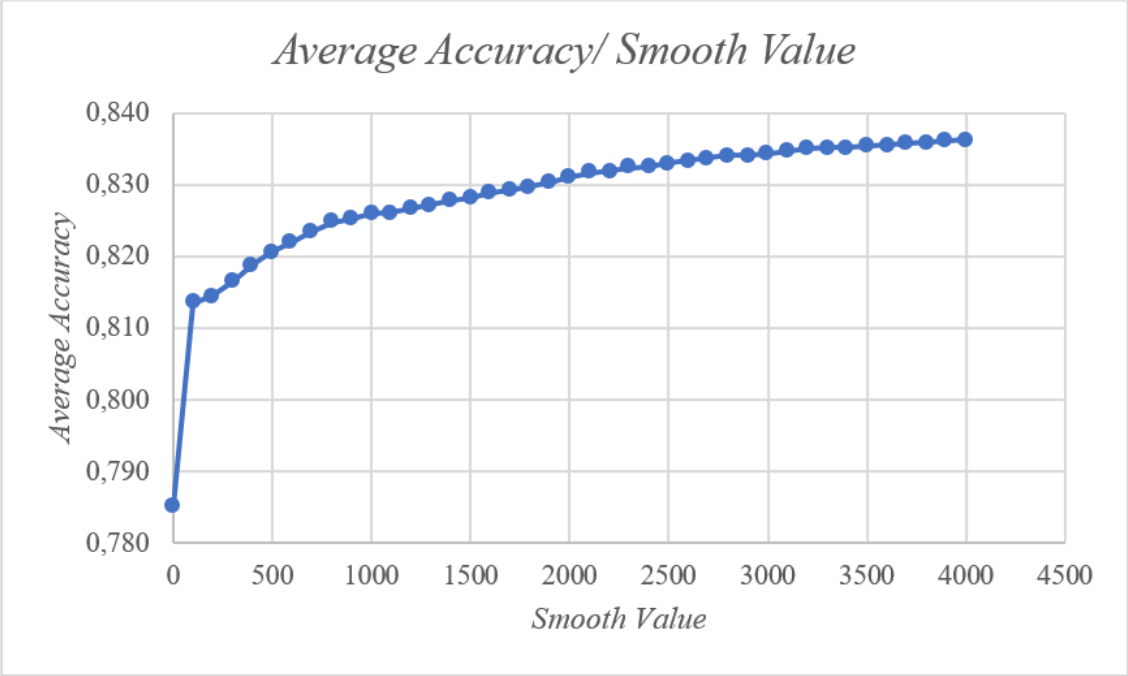

Figura B.29: ST10Test - Gráfico *Average Accuracy/ Smooth Value*

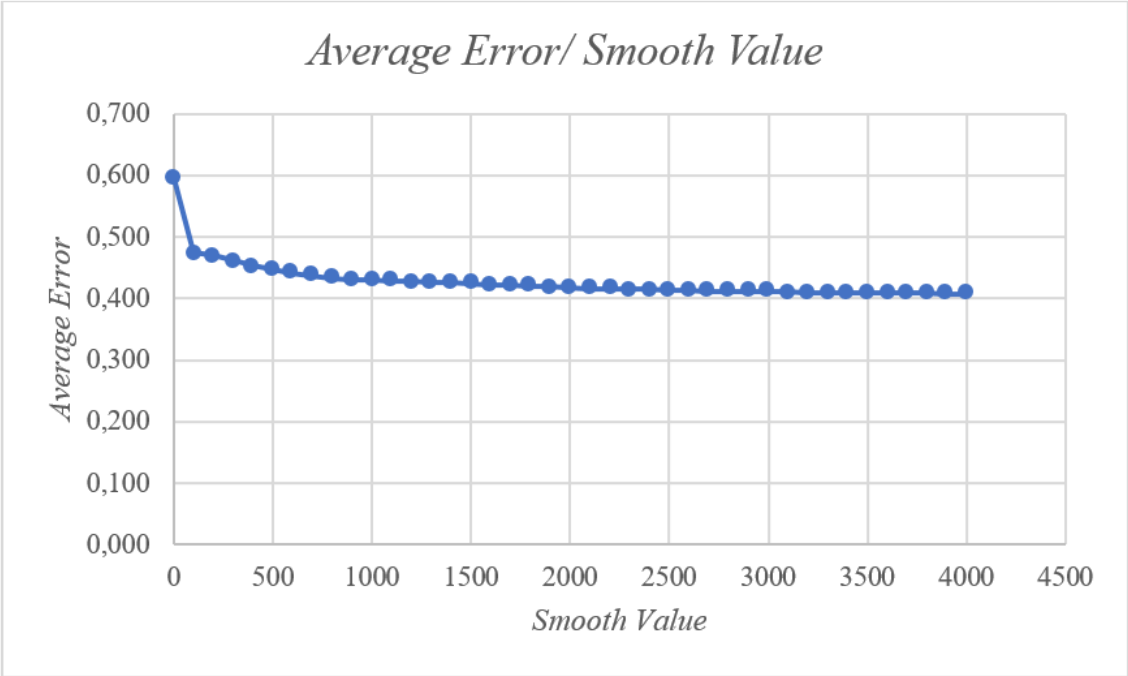

Figura B.30: ST10Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,051	0,125	0,215	0,785	0,596	1,11022E-16	0,072
100	0,285	0,300	0,187	0,813	0,474	0	0,292
200	0,263	0,291	0,186	0,814	0,469	1,11022E-16	0,276
300	0,266	0,295	0,184	0,816	0,462	5,55112E-17	0,280
400	0,271	0,300	0,181	0,819	0,453	5,55112E-17	0,285
500	0,274	0,303	0,179	0,821	0,447	0	0,288
600	0,278	0,306	0,178	0,822	0,441	5,55112E-17	0,291
700	0,282	0,308	0,177	0,823	0,437	5,55112E-17	0,295
800	0,281	0,309	0,175	0,825	0,433	5,55112E-17	0,295
900	0,283	0,309	0,175	0,825	0,431	5,55112E-17	0,296
1000	0,281	0,309	0,174	0,826	0,430	1,11022E-16	0,295
1100	0,286	0,309	0,174	0,826	0,429	0	0,297
1200	0,291	0,309	0,173	0,827	0,427	0	0,299
1300	0,292	0,308	0,173	0,827	0,426	0	0,300
1400	0,281	0,308	0,172	0,828	0,425	1,11022E-16	0,294
1500	0,291	0,308	0,172	0,828	0,424	0	0,299
1600	0,259	0,308	0,171	0,829	0,422	0	0,281
1700	0,257	0,307	0,171	0,829	0,421	5,55112E-17	0,280
1800	0,256	0,307	0,170	0,830	0,420	5,55112E-17	0,279
1900	0,258	0,307	0,170	0,830	0,419	0	0,280
2000	0,259	0,307	0,169	0,831	0,417	5,55112E-17	0,281
2100	0,260	0,308	0,168	0,832	0,416	0	0,282
2200	0,259	0,307	0,168	0,832	0,416	0	0,281
2300	0,259	0,307	0,168	0,832	0,415	5,55112E-17	0,281
2400	0,255	0,307	0,167	0,833	0,414	0	0,279
2500	0,260	0,307	0,167	0,833	0,413	5,55112E-17	0,282
2600	0,260	0,307	0,167	0,833	0,413	5,55112E-17	0,282
2700	0,257	0,307	0,166	0,834	0,412	5,55112E-17	0,280
2800	0,254	0,307	0,166	0,834	0,412	5,55112E-17	0,278
2900	0,241	0,306	0,166	0,834	0,411	0	0,270
3000	0,231	0,306	0,166	0,834	0,411	5,55112E-17	0,263
3100	0,230	0,306	0,165	0,835	0,411	0	0,263
3200	0,233	0,306	0,165	0,835	0,410	5,55112E-17	0,265
3300	0,236	0,306	0,165	0,835	0,409	0	0,267
3400	0,238	0,306	0,165	0,835	0,409	0	0,268
3500	0,241	0,306	0,165	0,835	0,409	0	0,270
3600	0,251	0,306	0,164	0,836	0,409	5,55112E-17	0,276
3700	0,255	0,306	0,164	0,836	0,408	5,55112E-17	0,278
3800	0,253	0,306	0,164	0,836	0,408	0	0,277
3900	0,258	0,306	0,164	0,836	0,407	0	0,280
4000	0,264	0,306	0,164	0,836	0,407	5,55112E-17	0,283

Tabela B.10: Resultados do teste ST10Test

ST11Test

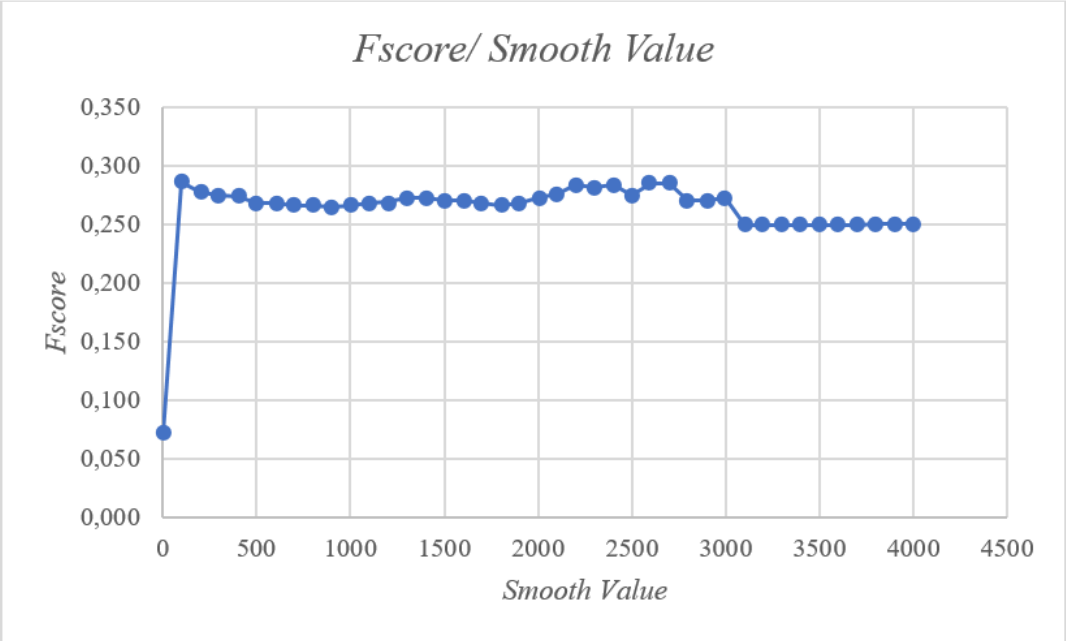

Figura B.31: ST10Test - Gráfico *Fscore/ Smooth Value*

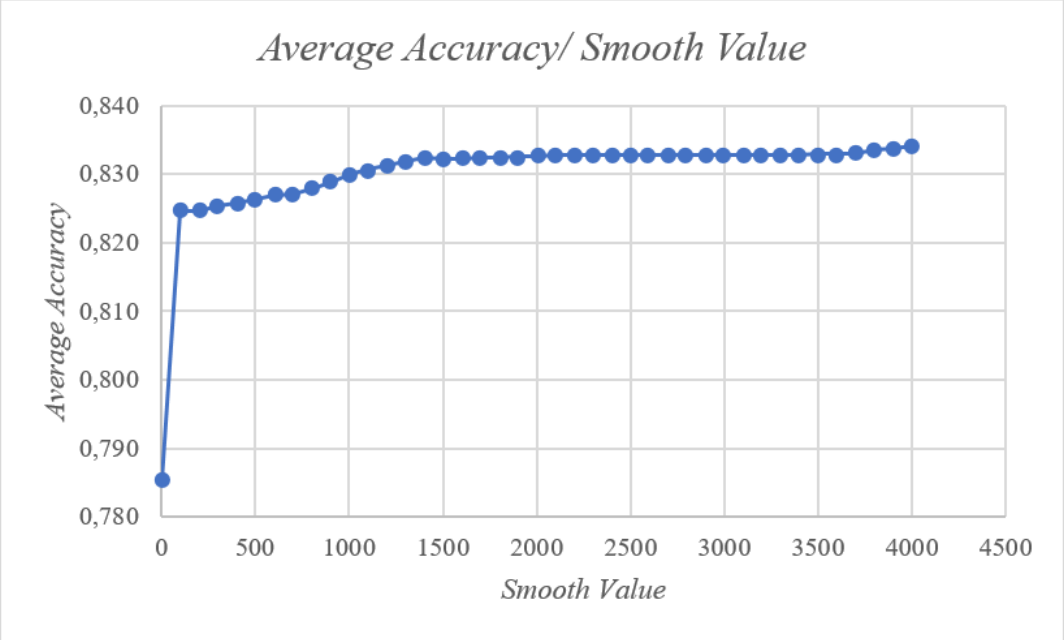

Figura B.32: ST11Test - Gráfico *Average Accuracy/ Smooth Value*

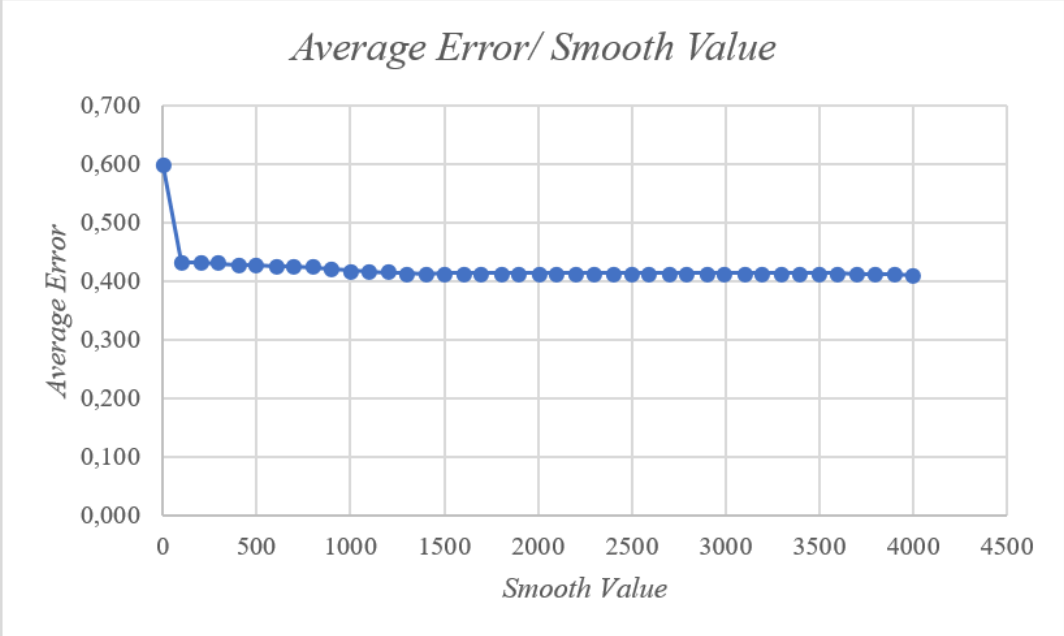

Figura B.33: ST11Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,050	0,125	0,215	0,785	0,596	1,11022E-16	0,072
100	0,267	0,308	0,175	0,825	0,432	5,55112E-17	0,286
200	0,253	0,307	0,175	0,825	0,432	0	0,277
300	0,248	0,307	0,175	0,825	0,430	5,55112E-17	0,274
400	0,246	0,308	0,174	0,826	0,428	5,55112E-17	0,274
500	0,238	0,308	0,174	0,826	0,427	0	0,269
600	0,238	0,308	0,173	0,827	0,425	5,55112E-17	0,269
700	0,234	0,308	0,173	0,827	0,425	1,11022E-16	0,266
800	0,234	0,308	0,172	0,828	0,423	5,55112E-17	0,266
900	0,232	0,309	0,171	0,829	0,421	0	0,265
1000	0,234	0,309	0,170	0,830	0,418	1,11022E-16	0,266
1100	0,236	0,309	0,169	0,831	0,416	5,55112E-17	0,268
1200	0,238	0,309	0,169	0,831	0,415	0	0,269
1300	0,244	0,310	0,168	0,832	0,414	0	0,273
1400	0,243	0,310	0,168	0,832	0,413	5,55112E-17	0,272
1500	0,241	0,309	0,168	0,832	0,414	0	0,271
1600	0,241	0,308	0,168	0,832	0,414	5,55112E-17	0,270
1700	0,237	0,308	0,167	0,833	0,413	0	0,268
1800	0,234	0,307	0,168	0,832	0,414	0	0,266
1900	0,238	0,307	0,167	0,833	0,414	0	0,268
2000	0,245	0,307	0,167	0,833	0,414	0	0,272
2100	0,250	0,307	0,167	0,833	0,413	5,55112E-17	0,276
2200	0,263	0,307	0,167	0,833	0,413	5,55112E-17	0,283
2300	0,260	0,307	0,167	0,833	0,414	0	0,281
2400	0,264	0,307	0,167	0,833	0,414	0	0,284
2500	0,249	0,306	0,167	0,833	0,414	0	0,275
2600	0,266	0,306	0,167	0,833	0,414	5,55112E-17	0,285
2700	0,266	0,306	0,167	0,833	0,414	5,55112E-17	0,285
2800	0,241	0,306	0,167	0,833	0,414	0	0,270
2900	0,241	0,306	0,167	0,833	0,414	0	0,270
3000	0,246	0,306	0,167	0,833	0,414	5,55112E-17	0,273
3100	0,210	0,306	0,167	0,833	0,414	0	0,249
3200	0,210	0,306	0,167	0,833	0,414	5,55112E-17	0,249
3300	0,210	0,306	0,167	0,833	0,414	5,55112E-17	0,249
3400	0,210	0,306	0,167	0,833	0,413	5,55112E-17	0,249
3500	0,210	0,306	0,167	0,833	0,413	0	0,249
3600	0,210	0,306	0,167	0,833	0,413	0	0,249
3700	0,210	0,306	0,167	0,833	0,413	5,55112E-17	0,249
3800	0,211	0,307	0,166	0,834	0,412	5,55112E-17	0,250
3900	0,211	0,307	0,166	0,834	0,411	0	0,250
4000	0,211	0,307	0,166	0,834	0,410	0	0,250

Tabela B.11: Resultados do teste ST11Test

ST12Test

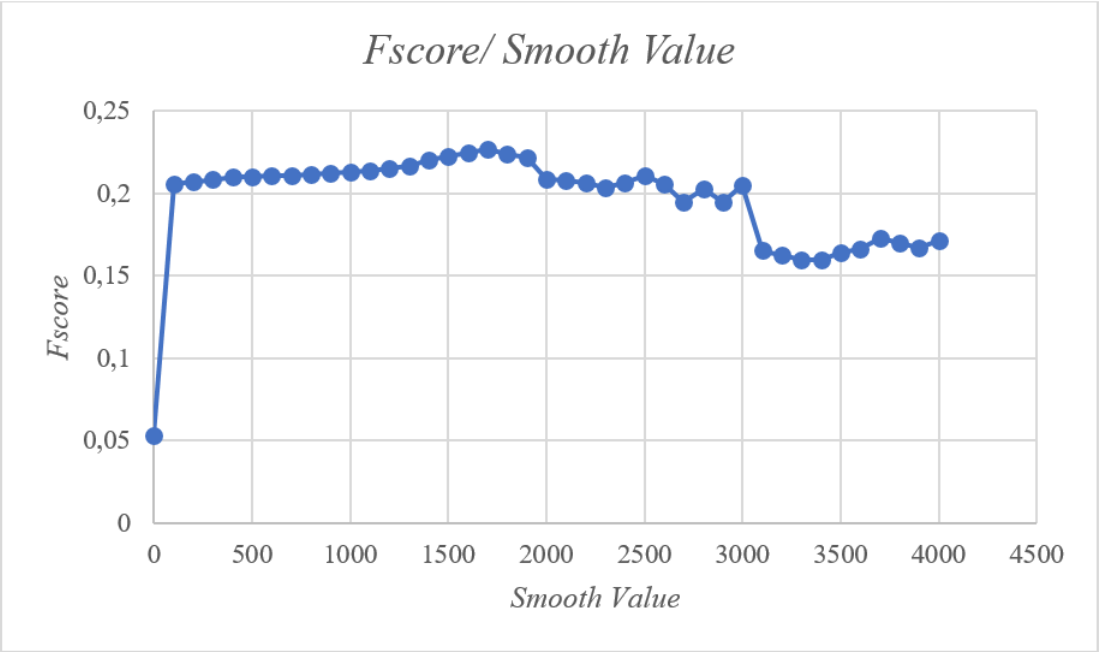

Figura B.34: ST12Test - Gráfico *Fscore/ Smooth Value*

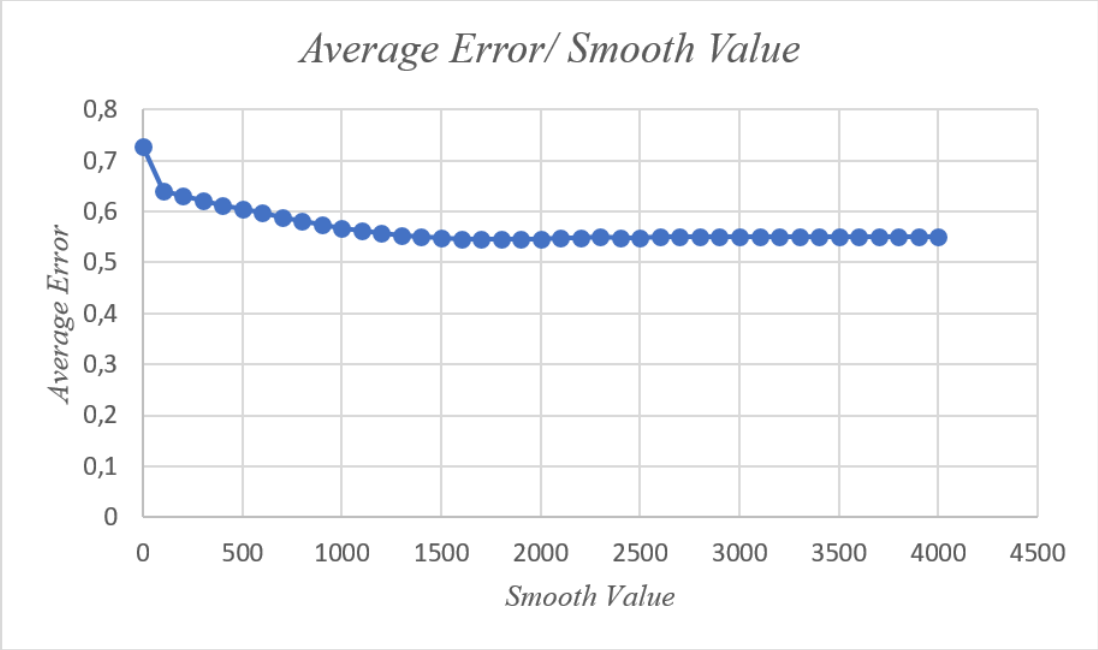

Figura B.35: ST12Test - Gráfico *Average Accuracy/ Smooth Value*

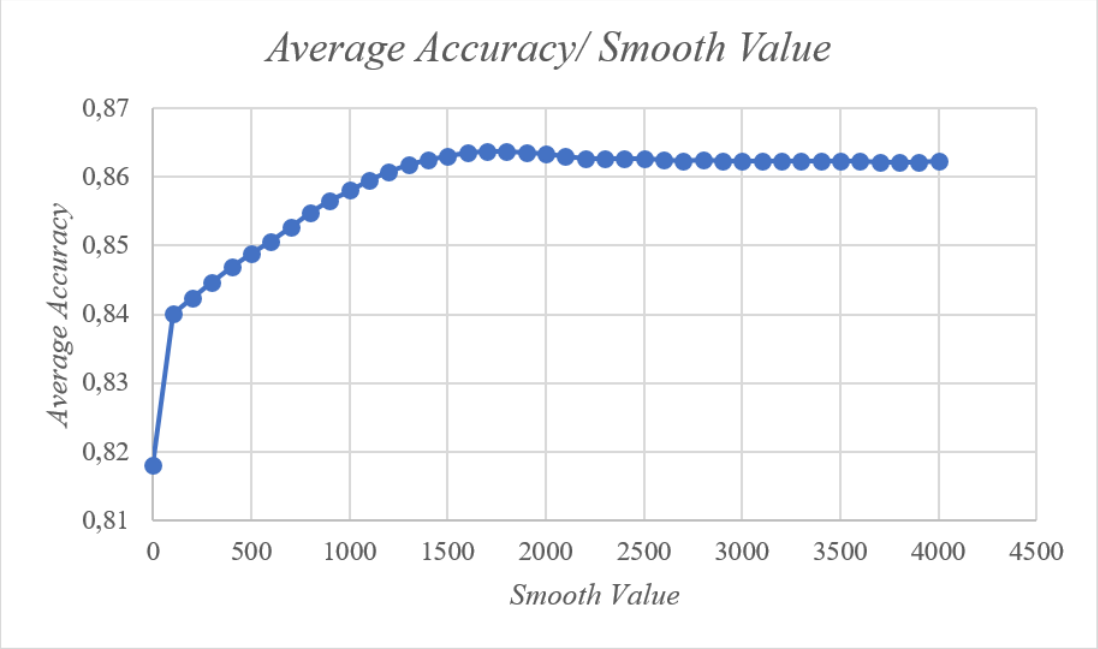

Figura B.36: ST12Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,034	0,125	0,182	0,818	0,728	1,11022E-16	0,054
100	0,203	0,209	0,160	0,840	0,639	1,11022E-16	0,206
200	0,204	0,211	0,158	0,842	0,630	1,11022E-16	0,207
300	0,204	0,213	0,155	0,845	0,621	1,11022E-16	0,208
400	0,204	0,216	0,153	0,847	0,612	1,11022E-16	0,210
500	0,203	0,217	0,151	0,849	0,605	0	0,210
600	0,203	0,218	0,149	0,851	0,597	1,11022E-16	0,211
700	0,203	0,220	0,147	0,853	0,589	1,11022E-16	0,211
800	0,203	0,222	0,145	0,855	0,581	0	0,212
900	0,202	0,223	0,143	0,857	0,574	0	0,212
1000	0,203	0,224	0,142	0,858	0,568	0	0,213
1100	0,203	0,225	0,141	0,859	0,562	0	0,214
1200	0,205	0,226	0,139	0,861	0,557	0	0,215
1300	0,208	0,226	0,138	0,862	0,553	1,11022E-16	0,217
1400	0,214	0,227	0,137	0,863	0,550	0	0,220
1500	0,220	0,226	0,137	0,863	0,548	0	0,223
1600	0,223	0,225	0,136	0,864	0,546	1,11022E-16	0,224
1700	0,229	0,224	0,136	0,864	0,545	1,11022E-16	0,227
1800	0,224	0,223	0,136	0,864	0,545	1,11022E-16	0,224
1900	0,223	0,221	0,137	0,863	0,546	1,11022E-16	0,222
2000	0,199	0,220	0,137	0,863	0,547	1,11022E-16	0,209
2100	0,199	0,218	0,137	0,863	0,548	1,11022E-16	0,208
2200	0,197	0,216	0,137	0,863	0,549	1,11022E-16	0,206
2300	0,193	0,215	0,137	0,863	0,550	1,11022E-16	0,203
2400	0,199	0,215	0,137	0,863	0,549	0	0,207
2500	0,207	0,215	0,137	0,863	0,549	1,11022E-16	0,211
2600	0,197	0,214	0,137	0,863	0,550	1,11022E-16	0,205
2700	0,179	0,214	0,138	0,862	0,550	1,11022E-16	0,195
2800	0,193	0,214	0,138	0,862	0,550	1,11022E-16	0,203
2900	0,179	0,213	0,138	0,862	0,551	1,11022E-16	0,194
3000	0,198	0,213	0,138	0,862	0,551	0	0,205
3100	0,135	0,213	0,138	0,862	0,551	0	0,165
3200	0,132	0,213	0,138	0,862	0,551	0	0,163
3300	0,128	0,213	0,138	0,862	0,551	0	0,160
3400	0,128	0,213	0,138	0,862	0,551	1,11022E-16	0,160
3500	0,133	0,213	0,138	0,862	0,551	1,11022E-16	0,164
3600	0,136	0,213	0,138	0,862	0,551	0	0,166
3700	0,146	0,212	0,138	0,862	0,551	1,11022E-16	0,173
3800	0,141	0,212	0,138	0,862	0,551	1,11022E-16	0,170
3900	0,137	0,212	0,138	0,862	0,551	1,11022E-16	0,167
4000	0,144	0,212	0,138	0,862	0,551	1,11022E-16	0,171

Tabela B.12: Resultados do teste ST12Test

ST13Test

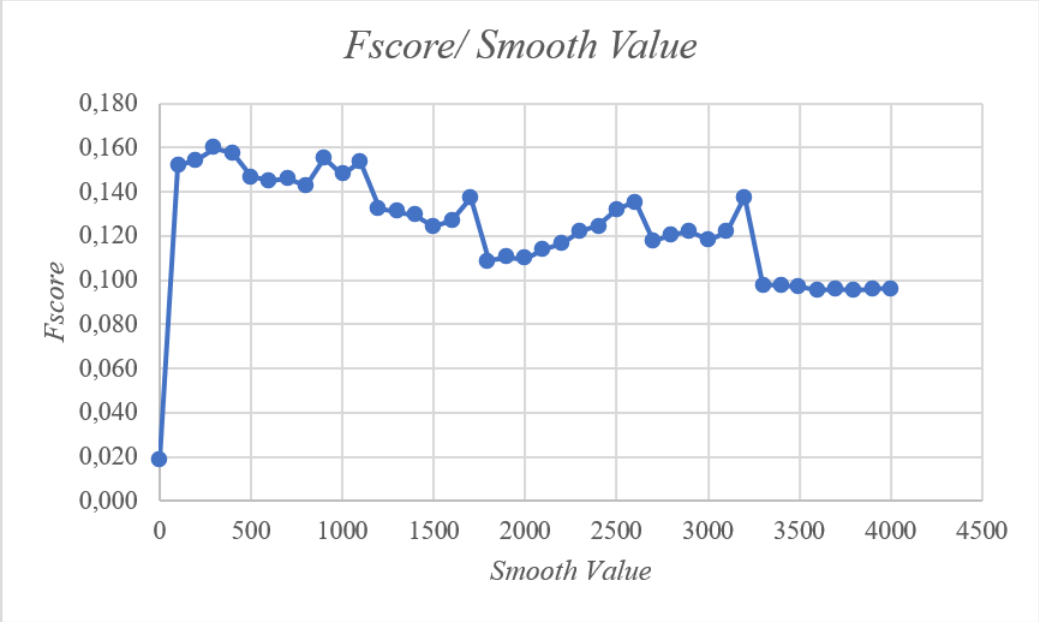

Figura B.37: ST13Test - Gráfico *Fscore/ Smooth Value*

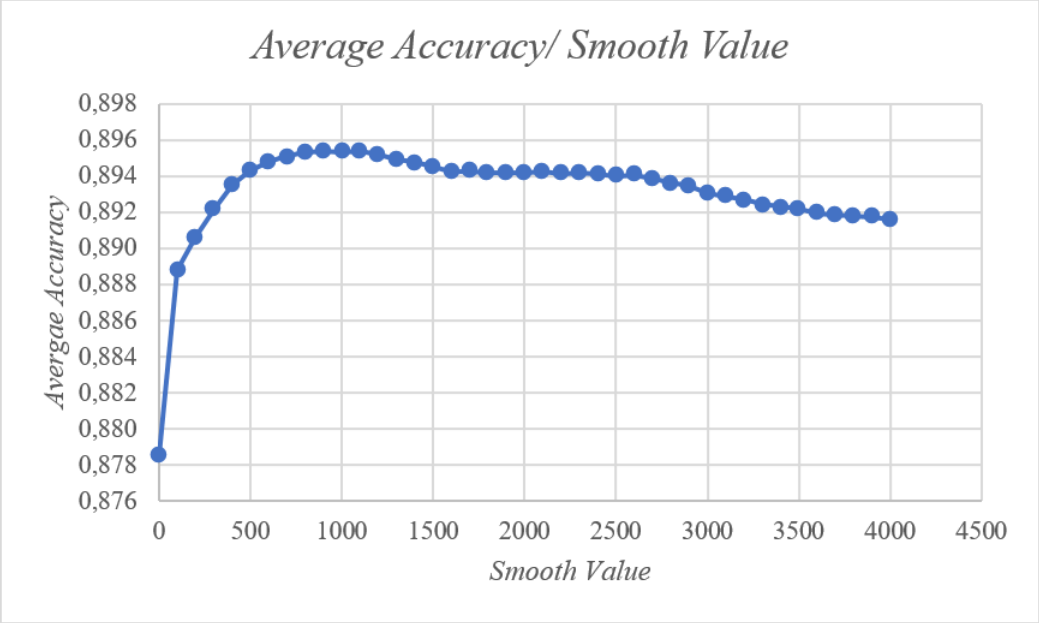

Figura B.38: ST13Test - Gráfico *Average Accuracy/ Smooth Value*

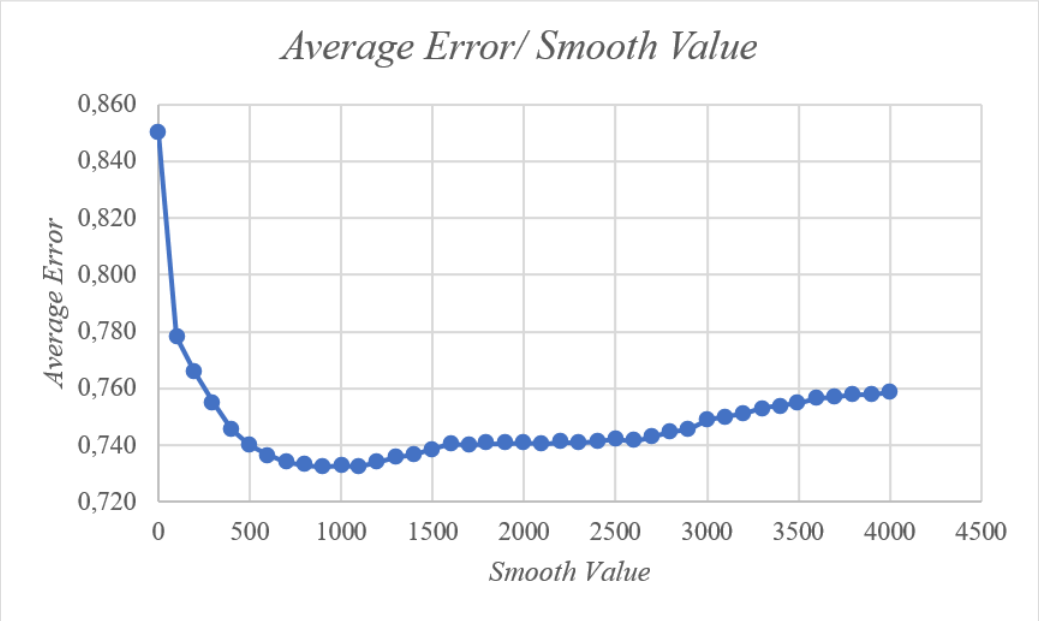

Figura B.39: ST13Test - Gráfico *Average Error/ Smooth Value*

Smooth Value	$Precision_M$	$Recall_M$	*Error Rate*	*Average Accuracy*	*Average Error*	*Standard Deviation*	$Fscore_M$
0	0,011	0,071	0,121	0,879	0,850	0	0,019
100	0,159	0,145	0,111	0,889	0,778	1,11022E-16	0,152
200	0,160	0,149	0,109	0,891	0,766	0	0,154
300	0,167	0,153	0,108	0,892	0,755	1,11022E-16	0,160
400	0,160	0,155	0,107	0,893	0,746	1,11022E-16	0,158
500	0,139	0,156	0,106	0,894	0,740	1,11022E-16	0,147
600	0,137	0,155	0,105	0,895	0,736	1,11022E-16	0,145
700	0,139	0,154	0,105	0,895	0,734	1,11022E-16	0,146
800	0,134	0,152	0,105	0,895	0,733	0	0,143
900	0,160	0,151	0,105	0,895	0,732	0	0,155
1000	0,147	0,148	0,105	0,895	0,733	0	0,148
1100	0,161	0,147	0,105	0,895	0,732	1,11022E-16	0,154
1200	0,122	0,145	0,105	0,895	0,734	1,11022E-16	0,132
1300	0,121	0,143	0,105	0,895	0,736	0	0,131
1400	0,120	0,141	0,105	0,895	0,737	0	0,129
1500	0,112	0,139	0,106	0,894	0,739	1,11022E-16	0,124
1600	0,118	0,137	0,106	0,894	0,740	0	0,127
1700	0,138	0,137	0,106	0,894	0,740	0	0,138
1800	0,091	0,136	0,106	0,894	0,741	1,11022E-16	0,109
1900	0,093	0,136	0,106	0,894	0,741	1,11022E-16	0,110
2000	0,092	0,136	0,106	0,894	0,741	1,11022E-16	0,110
2100	0,098	0,136	0,106	0,894	0,740	1,11022E-16	0,114
2200	0,103	0,135	0,106	0,894	0,741	0	0,117
2300	0,111	0,135	0,106	0,894	0,741	1,11022E-16	0,122
2400	0,116	0,134	0,106	0,894	0,741	1,11022E-16	0,125
2500	0,130	0,134	0,106	0,894	0,742	1,11022E-16	0,132
2600	0,138	0,134	0,106	0,894	0,741	1,11022E-16	0,136
2700	0,106	0,132	0,106	0,894	0,743	1,11022E-16	0,118
2800	0,111	0,131	0,106	0,894	0,745	0	0,120
2900	0,115	0,130	0,107	0,893	0,746	1,11022E-16	0,122
3000	0,110	0,128	0,107	0,893	0,749	1,11022E-16	0,119
3100	0,117	0,127	0,107	0,893	0,750	0	0,122
3200	0,152	0,126	0,107	0,893	0,751	0	0,138
3300	0,081	0,125	0,108	0,892	0,753	1,11022E-16	0,098
3400	0,081	0,124	0,108	0,892	0,754	1,11022E-16	0,098
3500	0,080	0,123	0,108	0,892	0,755	0	0,097
3600	0,079	0,122	0,108	0,892	0,756	1,11022E-16	0,096
3700	0,079	0,121	0,108	0,892	0,757	1,11022E-16	0,096
3800	0,079	0,121	0,108	0,892	0,758	1,11022E-16	0,096
3900	0,080	0,120	0,108	0,892	0,758	1,11022E-16	0,096
4000	0,081	0,120	0,108	0,892	0,759	0	0,096

Tabela B.13: Resultados do teste ST13Test

Abreviatura	Definição
AVI	Automated Vehicle Identification
AVC	Automated Vehicle Classification
BFFS	Base Free-Flow Speed
BN	Bayesian Network
DAG's	Directed Acyclic Graphs
DSRC	Dedicated Short Range Communications
EMEL	Empresa Municipal de Mobilidade e Estacionamento de Lisboa
ETC	Electronic Toll Collection
ETL	Extract, Transform, Load
FFS	Free Flow Speed
PHF	Peak Hour Factor
FND	Forma Normal Disjuntiva
FR	Flow Rate
HDFS	Hadoop Distributed File System
HOT	High Occupancy Toll
HOV	High Occupancy Vehicle
ID	Interchange Density
kNN	k-Nearest Neighbour
KPI	Key Performance Indicator
LC	Lane Cleareance
LMS	Least Mean Squared
LOS	Level Of Service
LW	Lane Width
ML	Machine Learning
MLlib	Machine Learning library
NB	Naive Bayesian
NoSQL	Not Only SQL
OCR	Optical Character Recognition
OLTP	Online Transaction Processing
PEI	Projeto de Engenharia Informática
RBF	Radial Basis Function
RES	Road Side Equipment
RFID	Radio Frequency Identification
RPC	Remote Procedure Call
RRD	Resilient Distributed Datasets
SAS	Statistical Analysis System
SVD	Singular Value Decomposition
SVMs	Support Vector Machines
VFDT	Very Fast Decision Tree
VHT	Vertical Hoeffding Tree